PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 17/60 — A1

(11) International Publication Number: WO 00/45319

(43) International Publication Date: 3 August 2000 (03.08.00)

(21) International Application Number: PCT/US00/02249

(22) International Filing Date: 28 January 2000 (28.01.00)

(30) Priority Data:
60/117,828 29 January 1999 (29.01.99) US

(71) Applicant: ONLINE INSIGHT, INC. [US/US]; 817 West Peachtree Street, Suite 600, Atlanta, GA 30308 (US).

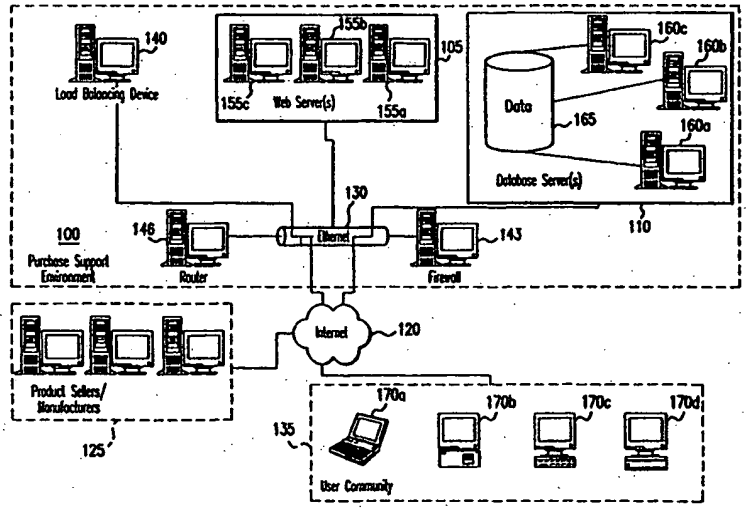
(72) Inventors: FORSTER, Kenneth, G.; 914 Virginia Avenue, Atlanta, GA 30306 (US). KREBS, Paul, E.; 914 Collier Road, #1304, Atlanta, GA 30318 (US). FLOWERS, Charlie; 1206 Parkwood Chase, N.W., Acworth, GA 30102 (US).

(74) Agents: KIRSCH, Gregory, J. et al.; Needle & Rosenberg, P.C., Suite 1200, 127 Peachtree Street, N.E., Atlanta, GA 30303 (US).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: MULTI-ATTRIBUTE SEARCHING SYSTEM AND METHOD FOR ELECTRONIC COMMERCE APPLICATIONS



(57) Abstract

The invention relates to a multi-attribute searching system and method for electronic commerce applications. This system and method provides purchase decision support to a purchaser with respect to a product type. A typical system includes a data store (110) for storing user preferences and a server (105) to interact with a purchaser (135), generate user preferences and provide purchase recommendations. The present invention provides a virtual salesperson that tailors its interaction for each individual consumer. A system according to the present invention educates, asks questions, interprets responses and presents recommendations (370). Further, a detailed preference profile for each individual consumer is collected and stored.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

MULTI-ATTRIBUTE SEARCHING SYSTEM AND METHOD FOR ELECTRONIC COMMERCE APPLICATIONS

CROSS-REFERENCE TO RELATED PATENT APPLICATION

This application claims the benefit, pursuant to 35 U.S.C. § 119(e), of applicants' provisional U.S. Patent Application Serial No. 60/117,828, filed January 29, 1999, entitled "MULTI-ATTRIBUTE SEARCHING SYSTEM AND METHOD FOR ELECTRONIC COMMERCE APPLICATIONS".

BACKGROUND OF INVENTION

1. FIELD OF INVENTION

The present invention relates to a multi-attribute searching system and method for electronic commerce applications. More specifically, the present invention relates to a system and method for generating individually-valid, statistics-based user preferences, via a purchaser specific interview, which are used to provide purchase recommendations. Generated user preferences are stored for subsequent analysis and use at the individual and aggregate level.

2. DESCRIPTION OF PRIOR ART

The Internet is a global network of connected computer networks. Over the last several years, the Internet has grown in significant measure. A large number of computers on the Internet provide information in various forms. Anyone with a computer connected to the Internet can potentially tap into this vast pool of information.

The most wide spread method of providing information over the Internet is via the World Wide Web (the Web). The Web consists of a subset of the computers connected to the Internet; the computers in this subset run Hypertext Transfer Protocol (HTTP) servers (Web servers). The information available via the Internet also encompasses information available via other types of information servers such as GOPHER and FTP.

Information on the Internet can be accessed through the use of a Uniform Resource Locator (URL). A URL uniquely specifies the location of a particular piece

of information on the Internet. A URL will typically be composed of several components. The first component typically designates the protocol by with the address piece of information is accessed (e.g., HTTP, GOPHER, etc.). This first component is separated from the remainder of the URL by a colon (':'). The remainder of the URL will depend upon the protocol component. Typically, the remainder designates a computer on the Internet by name, or by IP number, as well as a more specific designation of the location of the resource on the designated computer. For instance, a typical URL for an HTTP resource might be:

http://www.server.com/dir1/dir2/resource.htm

where http is the protocol, www.server.com is the designated computer and /dir1/dir2/resouce.htm designates the location of the resource on the designated computer.

Web servers host information in the form of Web pages; collectively the server and the information hosted are referred to as a Web site. A significant number of Web pages are encoded using the Hypertext Markup Language (HTML) although other encodings using the eXtensible Markup Language (XML) or the Standard Generic Markup Language (SGML) are becoming increasingly more common. The published specifications for these languages are incorporated by reference herein. Web pages in these formatting languages may include links to other Web pages on the same Web site or another. Web servers, information servers of other types, await requests for the information that they host from Internet clients.

Client software has evolved that allows users of computers connected to the Internet to access this information. Advanced clients such as Netscape's Navigator and Microsoft's Internet Explorer allow users to access software provided via a variety of information servers in a unified client environment. Typically, such client software is referred to as browser software.

As these technologies have evolved, the application of marketing research and statistical techniques to the problem of modeling consumer purchasing behaviors has also evolved greatly over the past several decades, but has been limited by several problems, including a failure to appropriately model real-life purchase decisions, a

failure to customize the research task to the individual and the particular decision event, and a failure to fully exploit technological developments.

Standard approaches have involved consumer surveys, with questions designed to ask the consumer to state the importance of various features to their decision to buy a particular product. Several methodologies such as conjoint analysis have been developed to improve upon these efforts, using advanced statistical approaches to model the trade-offs consumers make between these features and thereby better understand how different products might compare to one another. By collecting this information, conjoint has traditionally been used for new product development, pricing studies, customer segmentation, and targeting of marketing messages. This early development was extended with more sophisticated computer-assisted interviews that were adaptive to the individual respondent with a technique called adaptive conjoint analysis (ACA), first commercialized by Sawtooth Software, Inc.

Researchers later began to use a conjoint approach in a computer-assisted environment for decision support, but this was not known to be used in commercial applications due to the fact that these systems failed to exploit the real-time interactive and distributed nature of client-server systems and Internet technologies.

At the same time, the application of advanced analysis and profiling to the e-commerce space has been severely limited. Many companies have focused efforts on tracking consumer behaviors online to infer attitudes and preferences, but not have applied the analytical rigor of a statistically-valid and predictive approach, such as conjoint, to the Internet. The application of the advanced analytics of conjoint analysis on the Internet to aid in e-commerce purchase decisions is a new one. In order for conjoint to be effective in this medium, significant changes need to be made as traditional conjoint analysis is not user friendly, requires too many questions to be asked to model user preferences, and requires that all respondents submit to the same interview process.

Instead of conjoint, many online companies have relied on systems such as collaborative filtering that infer an individual's purchasing behavior from the behaviors of the 'like-minded' consumer cluster to which they are assigned, or on systems that allow users to query and/or filter a database eliminating product or service options that

do not meet specific criteria. The problem with these approaches is that they are neither unique nor explainable to the individual in the case of the former, and may result in two many or too few products being eliminated out of the consideration set, and a need for consumers to understand the terms of the database constraints in the case of the latter.

These systems have also suffered from deficiencies as it relates to the customizability of the interview. This relates both to the ability for each interview to be dynamically adapted to the individual respondent, the product being evaluated, and/or the merchant offering the product, and to the ability for the user interface to be customized in any manner deemed appropriate by the merchant.

Existing systems have not yet addressed these functional deficiencies nor have they fully exploited the possibilities of existing technologies to provide the maximum business benefits possible.

SUMMARY OF THE INVENTION

The present invention is directed to a system and method for multi-attribute searching for electronic commerce applications. According to the present invention, a purchaser may receive automated purchase decision support.

The present invention provides the e-commerce equivalent of an expert salesperson in the traditional buying environment. In a preferred embodiment, a system according to the present invention includes a server for interacting with the user and generating preference profiles in communication with a data store for storing preference profiles. Interactions occur with a consumer to help him or her find the right solution for a purchasing decision. The server according to the present invention typically is responsible for educating the consumer, asking questions of the consumer, analyzing responses, and presenting product recommendations potentially with explanations of the recommendations both overall and feature by feature.

The best salespeople learn everything they can about their customers and record this information so that they can serve them more effectively in the future. For each consumer, a preference profile that expresses and quantifies how this person makes purchasing decisions is built and stored. During an interview, the processor walks a

consumer through a question and answer process. At the start of each interview, the server accepts a definition of the attributes that define the decision being made. This definition of attributes is called the study definition, or the study for short. Because the server can accept this information at the start of each interview, a single implementation can guide consumers through any number of different decisions.

The number of questions and types of attributes can be different for every consumer, so novices can be treated differently than pros. A consumer can even specify which attributes matter and which attributes do not matter, effectively controlling the kinds of questions the server asks in the interview.

To provide maximum flexibility, all information input into or output from the server is communicated in an XML format. This allows ease of integration between the server and most any other software system.

Note also that at the start of each interview a product catalog may be provided to the server. As with all other forms of input, this catalog is specified in terms of XML. Specifying this information at the start of each interview results in a tremendous amount of flexibility. Some products may not be available to certain consumers, or certain special products may be available only to select individuals.

The above and other objects and advantages of the present invention will become more readily apparent when reference is made to the following description, taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagram of the components in a typical system according to the present invention.

FIG. 2 is an example of a bar graph displaying relative importance of attributes as displayed to a user.

FIG. 3 is a flowchart of a process according to the present invention.

FIG. 4 is a flowchart of a process to calculate priors utilities.

FIG. 5 is a flowchart of a process to calculate pairs utilities.

FIG. 6 is a diagram of the communication between the presentation layer and the analysis engine of the preferred embodiment.

FIGs. 7-20 are sample screen captures.

FIG. 21 is a block diagram depicting organization of clusters.

FIG. 22 is a block diagram of the structure of an adaptor.

FIGs. 23-29 are diagrams illustrating data flow from clients to a cluster.

FIG. 30 is a diagram of the architecture for an adaptor handling CORBA objects.

DETAILED DESCRIPTION OF THE INVENTION

A preferred embodiment of the invention is now described in detail. Referring to the drawings, like numbers indicate like parts throughout the views. As used in the description herein and throughout the claims that follow, the meaning of "a," "an," and "the" includes plural reference unless the context clearly dictates otherwise. Also, as used in the description herein and throughout the claims that follow, the meaning of "in" includes "in" and "on" unless the context clearly dictates otherwise. Further, the terms defined in the following definitions section shall apply in the description herein and throughout the claims that follow.

Definitions

Study: A study encapsulates the set of attributes, the levels for each attribute, and the parameters that determine the nature of an interview.

Respondent: A respondent is a participant in an interview.

Purchaser: This term shall be construed as synonymous with respondent or consumer and shall include both actual and potential purchasers of products.

Product: This term shall be construed to cover both goods and services.

Study Parameters: Configurable properties of a study that determine the nature of the study. For example, interview time limit.

Attribute: An attribute is a feature of the product being studied which can have various levels. For example, for a study involving cars, an attribute might be "Color".

Attribute Level: Each attribute involved in a conjoint study can have many different levels. For example, for a study involving cars, an attribute might be "Color", and attribute levels might be "Red", "Black", "Yellow", and "Green".

Pairs: A type of question asked during the interview, which presents two profiles and asks the respondent to indicate her preference for one or the other on a scale ranging from "Strongly prefer left" to "Strongly prefer right". Note: The term "Pairs" can also refer to the section within the interview where the Pairs questions are asked.

Priors: A section of the interview that occurs before the Pairs section. Several different types of questions may be asked in this section, including, for example, Ranking and Importance questions.

Utilities: Numbers that represent the value that a respondent places on each attribute level. There is one utility value per respondent per attribute level. To determine the relative desirability of an entire product profile, the utilities of the attribute levels which make up the product profile are summed. There may be two sets of utilities per respondent: priors utilities and pairs utilities (both defined below).

Priors Utilities: The utility values that the system calculates based on the respondent's responses in the Priors section of the interview.

Pairs Utilities: The utility values that the system calculates based on the respondent's responses in the Pairs section of the interview.

Unacceptables: An optional section of the interview that can be used to eliminate certain attribute levels if the interview would otherwise be too long. The respondent is shown a list of levels for each attribute and asked to eliminate any that would be completely unacceptable.

Preference Ranking/Rating: A question within the interview in which the respondent is asked to indicate her preference for various attribute levels. The study can be configured to allow respondents to rank the attribute levels in order of desirability or to allow respondents to rate the desirability of each attribute level. This question can be omitted for those attributes where the preference should be obvious (such as price).

Ranking: A method by which the respondent can indicate his or her preference for various attribute levels. The ranking method does not allow for ties, and therefore the respondent must assign each level a unique numerical value indicating his or her preference, with lower numbers indicating a higher preference. Since ties are not allowed, the scale of this response is between 1 and the total number of levels for the attribute.

Rating: A method by which the respondent can indicate his or her preference for various attribute levels. The rating method allows for ties, and therefore the respondent must assign each level a numerical value indicating his or her preference, with lower numbers indicating a higher preference. Since ties are permitted, the scale of this response is arbitrary.

(The number of attribute levels which can be considered during the interview is a Study Parameter which the study-planner can set.)

Importance Ratings: A section of the interview which asks the respondent to rate the importance of each attribute to his or her purchase decision. I n a preferred embodiment, the system shows the respondent the best level and the worst level for an attribute and asks him or her how important the difference between the two levels is. For example, in a study concerning cars, an example question would ask "If 2 new cars you were considering were both acceptable except for the SINGLE DIFFERENCE shown below, how important would THAT DIFFERENCE be?...A) Made in USA; B) Made in Japan."

Calibration Concepts: A section in the interview where the system shows the respondent a set of product profiles and, for each profile, asks the respondent to enter a percentage (from 0 to 100) expressing how likely the respondent would be to buy that particular product. The results are used to scale the respondent's utilities, and may be used to determine how to weight the priors utilities versus the pairs utilities. Calibration is optional

Preliminary Questions: This term refers to the following sections of the interview: Unacceptables, Preference Ranking/Rating, and Importance Ratings.

a priori Levels: Some attribute levels have a natural order of desirability that would be the same for all (or nearly all) respondents. For example, all else being equal, a price of $10,000 is more desirable than a price of $12,000. These kinds of attribute levels are called a priori levels. The study-planner can specify that any attribute's levels have a priori desirability, either in increasing or decreasing order.

Prohibited Pairs: Some attributes or attribute levels should never appear in the same question as other attributes or attribute levels. For example, if a study is exploring different levels of "Dollars per gallon" and "Dollars per case", the study-planner would

want to prohibit questions concerning both of these attributes from occurring in the conjoint interview.

Constraints: Constraints refer to hard limits set on the attributes in a study. This includes unnacceptables as well as 'must haves' (i.e. this laptop must have a 300mhz processor).

Equal Weighting: The first of two possible methods for weighting the priors utilities versus the pairs utilities. This method places equal weight on the priors utilities and the pairs utilities.

Optimal Weighting: The second of two possible methods for weighting the priors utilities versus the pairs utilities. This method used the responses from the calibration questions to determine what weightings to apply.

Simulation: When a study-analyst has gathered the results of a study, she can run simulations against it to determine the predicted success of various potential products. A simulation consists of a Base Case Product (defined below) and/or any number of Scenarios (also defined below). The study analyst can select one of three choice models to apply to the base case and scenarios in the simulation: Maximum Likelihood, Share of Preference, and Purchase Likelihood. Each of these is defined below.

Maximum Likelihood: A choice model that specifies that a simulation should allocate each respondent's preference entirely to the one product that has the highest utility for that respondent.

Share of Preference: A choice model that specifies that a simulation should allocate preferences fractionally, taking into account the respondent's utilities for each product.

Purchase Likelihood: A choice model which specifies that a simulation should estimate the probability that each specific respondent will purchase each product.

Base Case Product: A set of simulation parameters that defines the standard product profile. When a simulation is run against the Base Case Product, the system will calculate various predictions about the relative success of the product profiles.

Scenario: A set of simulation parameters and a list of product profiles. When a simulation is run against the Scenario, the system will calculate various predictions about the relative success of the product profiles.

Generation of User Preferences

Conjoint analysis, or simply conjoint, is a robust statistical technique that supports learning how people make decisions about goods or services that are made up of a several different features. For example, for computers, features may include RAM, hard drive capacity, processor speed, screen size, etc. Conjoint is used in the preferred embodiment of the present invention to generate user preferences. In other embodiments, other forms of individually-valid statistical analysis may be used.

Conjoint utilizes a questioning method that mimics the real world by showing consumers products, each composed of several features, and letting them indicate how much they like them. By varying the features of the products and analyzing responses, it is possible to quantify how each aspect of a product drives preference. As details of how much value people place on each possible feature are acquired, these value components can be added to predict how much the purchaser would like a potential product offering. This allows the comparison of many possible products and uncovers the best fit for the consumer.

There are several steps involved in the adaptive conjoint process **300** of the preferred embodiment of the present invention. These steps are seen in FIGs. 3-5, outlined and described more fully below:

1) Defining attributes and levels may optionally occur **310**
2) The Priors Conjoint Section may optionally occur **320**
   - Ranking or Rating of levels within attributes may optionally occur **410**
   - Rating of attribute importance may optionally occur **420**
   - Setting attribute constraints **430** and identifying unacceptables may optionally occur. In the preferred embodiment this will occur in conjunction with the rating of attributes and levels, but in other embodiments this may occur either before or after the rating of attributes and levels.
   - Calculating priors only utilities **440**
3) The Pairs Conjoint Section may optionally occur **330**
   - Determining the number of pairs questions **510**
   - Prohibiting pairs may occur **520**
   - Choosing pairs **530**

- Calculating pairs only utilities **540**
- Calculating equal weight utilities **340**

4) Calibration Concepts may optionally occur **350**

- Calculating final optimal weighted utilities

5) Calculating of final attribute relative importance **360**

6) Product Scoring and Recommending based on User Preference Profile(s) **370**

Example Used Throughout

All of the calculations required will be explained based upon the example provided below. The example study tests on 3 attributes with a total of 9 levels (4 levels in Attribute 1, 3 levels in Attribute 2, and 2 levels in Attribute 3).

The example product type is laptop computers, and names have been added to the attributes and levels below for clarity:

Attribute #	Attribute Name	Level #	Level Name
1	*Brand*	1	Dell
		2	Compaq
		3	IBM
		4	Gateway
2	*Price*	5	$3,000
		6	$2,250
		7	$1,500
3	*Processor*	8	Pentium
		9	Pentium II

Calculating Conjoint Priors

The 'priors utilities', or simply 'priors' are calculated from two pieces of information given by the user – their ranking or rating of levels within each attribute

and their rating of the relative importance of each attribute. Priors are the initial set of preferences used to shape and validate the questions in the next section, the conjoint pairs. In a preferred embodiment, the calculation of priors occurs optionally.

Priors calculation process is described more fully below.

Attributes and Levels

Users will be explicitly instructed that they will see a series of product features, known in conjoint analysis parlance as attributes. Consumers will be able to click on any product feature and receive a detailed explanation of the feature and how it may or may not influence their decision-making process. The ability to learn about a specific product feature is an important aspect of the present invention.

There are some guidelines to be followed as to the numbers of attributes and levels that can be included:

- Total number of attributes cannot be less than two or greater than some fixed maximum
- The number of levels in an attribute cannot be less than two or greater than some fixed maximum.
- Users may be able to place constraints on the levels that are/are not included in the study/search. This is discussed in more detail below.

Self-explicated Rankings of Levels with Attributes

This is the first actual conjoint task introduced to consumers – the ranking or rating of attribute levels. Once they fully understand the product features that they will be interacting with, the consumer will be presented with a list of levels for each product feature.

If the ranking approach is employed, they will be asked to rank these levels in terms of their desirability or attractiveness from 1 to n, with n being the number of levels for the particular attribute.

For many product attributes, the order of the levels can be predetermined (ranked *a priori*) and does not need to be asked directly of the consumer. An example is price, where if all other things are considered to be equal people almost universally prefer a lower price to a higher one. Even if *a priori*, the attribute will still need to be included in the relative importance rankings described in the next section.

Calculation Process for Priors Ranks

As mentioned above, users are asked to rank each of the levels within each attribute. The rankings given by the users are recorded as a series of numbers. Each number is the answer given by the web visitor and is an ordinal number – 1 for the first level chosen, 2 for the second, etc. Next the rankings are reversed – the lowest rank becomes the highest; the highest becomes the lowest, etc. So, if there are five levels, the 1 becomes 5 and 2 becomes 4 and so on. Finally, the ranked values for each level are centered at zero for each attribute, using the rules below:

- Values for levels are to be centered at zero
- Values for levels are to be evenly spaced with a unit value of one between each level
- The computation to center values at zero is:

  New Value = Reversed rank–Average rank of levels in the attribute

The average rank of levels in an attribute is as follows:

Number of Levels in the Attribute	**Rule**
2 levels	1.5
3 levels	2
4 levels	2.5
5 levels	3
And so on....where Average = (Number of Levels+1)/2	

After the first stage of ranking the levels within each attribute the values should look as follows (assuming the user provided the rank detailed in the 'Rank Given' field below):

Attribute Number	**Level Number**	**Rank Given**	**Reversed Rank**	**Centered At Zero**
1	1	1	4	1.5
	2	2	3	.5
	3	3	2	-.5

	4	4	1	-1.5
2	5	3	1	-1
	6	2	2	0
	7	1	3	1
3	8	1	2	.5
	9	2	1	-.5

Note: the spread between the highest and lowest values within each attribute is equal to the number of levels within that attribute minus one.

Rating of Attribute Relative Importance

From the ranking questions, the order of levels will now have been established for each product feature. The consumer must then indicate how important each attribute is to his or her purchase decision. In the preferred embodiment, they will do this by indicating how important the difference between their most preferred level and their least preferred level is for each attribute. The stronger the importance between the highest and lowest level, the more important the attribute is to the consumer. In a preferred embodiment, the rating of attribute relative importance may occur concurrently with the ranking or rating of the levels within the attributes.

The importance rating for each attribute will be the answer (1,2,3, or 4) given by the user. In the example, let's suppose that the following importance ratings were given:

Attribute	**Attribute Name**	**Importance Rating**
1	*Brand*	3
2	*Price*	4
3	*Processor*	1

Adjusting Prior Ranks for Attribute Importance

For each attribute, the zero-centered values from the ranking section must be adjusted so that the range from the highest to the lowest value equals the importance

rating for that attribute. Also, the difference between all adjacent values within an attribute must be a constant.

Example: For Attribute 2 the importance rating is 4. So the range from the high to the low value must also be 4. Centered at zero, this implies a high of 2 and a low of –2. The adjusted values for the three levels in Attribute 2 will then be -2, 0 and 2. Note that the difference between adjacent values is the same.

The following rules can be used to make these adjustments in the rank values:

- The highest value is 1/2 of the importance rating and the lowest value is the negative of the highest value. This will ensure that the range is equal to the importance rating.
- If there are an odd number of levels, the middle value is 0.
- If there are 4 levels, the second highest value is 1/3 of the highest value and the second lowest value is the negative of the second highest.
- And so on...With the general formula being that the Adjusted Prior Utility equals:

(Centered at Zero value)/(n-1)*(Importance Ranking) where n equals the number of levels in the attribute.

Attribute Number	**Level Number**	**Rank**	**Reversed Rank**	**Centered At Zero**	**Importance Ranking**	**Adjusted Prior Utilities**
1	1	1	4	1.5	3	1.5
	2	2	3	.5	3	.5
	3	3	2	-.5	3	-.5
	4	4	1	-1.5	3	-1.5
2	5	3	1	-1	4	-2
	6	2	2	0	4	0
	7	1	3	1	4	2
3	8	1	2	.5	1	.5

	9	2	1	-.5	1	-.5

The final column in the table above is the vector of values of the dependent variable to be used in the regression equations later. These are the prior utilities for the nine example levels.

Calculation Process for Priors Ratings

The ranking approach to calculating priors level utilities is described above, and the ratings approach is defined as follows. The advantage of ratings over rankings is that it allows ties –i.e. allows two levels to be equally preferable, and also allows the user to state the absolute attractiveness of each given level – i.e. even with the highest relative rating, a level can be given a high rating or low rating, the latter indicating that none of the levels are particularly attractive.

The calculation of Prior Utilities depends on the type of question being posed. If the preference order of levels within an attribute can be reasonably inferred, the attribute is considered an a priori attribute. That is, the level of importance within the attribute can be defined at the beginning of the study. An example of an a priori attribute is price. The attribute price refers to the product cost, while the levels within price represent the various possible prices for the product. Price is a priori because one can reasonably assume that lower prices will always be preferred by consumers.

Using a ratings approach, Priors Utilities are calculated differently for a priori and non-a priori attributes as defined below.

A Priori Ratings-based Priors Utility

A priori levels are treated similarly to the standard ranking approach. For each level within the attribute, create a rank from one to n, where n is the total number of levels within the attribute. A rank of one is assigned to the level within the attribute that is most preferred and n is assigned to the least preferred level. Assume the a priori attribute is price and that price has four levels: $1000, $2000, $3000and $4000. The price of $1000 receives a rank of one, $2000 a rank of two, etc.

Next, reverse the rankings so that the most preferred level receives the highest rank. Thus, $1000 receives a four, $2000 receives a three, etc.

Third, adjust the reversed ranks so they are centered at zero. This is done by subtracting the following number from each reversed rank: the sum of the ranks divided by the number of levels in the attribute. In the example having four levels, 2.5 is subtracted from each reverse rank ({4+3+2+1} / 4).

Finally, calculate the Prior Utility using the following equation:

Prior Utility = (zero-centered rank*importance) / (number of levels in attribute – 1)

Importance is a number from one to four provided by the interviewee – four being very important and one being very unimportant.

Non-A Priori Ratings-based Priors Utility

For variables where preference order may not be assumed up front, the ratings method, which differs from rankings in its calculations, is used to derive Priors Utility. In addition to providing attribute importance, the user also provides a rating for each level within the attribute using a scale from one to five – five representing a very attractive level and one representing a very unattractive level. For our example, assume the interviewee is rating three laptop computer brands: Dell, Compaq, and IBM. Assume the interviewee provided the following ratings:

Level	Rating
Dell	5
Compaq	3
IBM	2

First, adjust the ratings so they are zero-centered on a five point scale. Thus, possible scores will range from –2 to +2, rather than one to five. This is done by subtracting the following number from each rating: the sum of the numbers in the rating scale divided by the number of levels in the rating scale. For a one to five scale, three is subtracted from each rating ({5+4+3+2+1} / 5).

Then, calculate Prior Utility using the following equation:

Prior Utility = { (zero-centered rating*importance) / spread of interviewee ratings } –
{ sum of zero-centered ratings / number of levels in attribute }

Priors Matrix

These utilities are used to set up the priors matrix. This matrix will be updated in the pairs section and used to calculate pairs utilities as well as to determine the pairs

to be asked. At the beginning the matrix is set to be the identity matrix for the nine levels (independent variables) in the study and the prior utilities are set as the dependent variables.

Priors Utility Matrix

		Independent X variables (levels)										
Attribute	**Level**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**Dependent Y**
1	1	1	0	0	0	0	0	0	0	0		1.5
	2	0	1	0	0	0	0	0	0	0		.5
	3	0	0	1	0	0	0	0	0	0		-.5
	4	0	0	0	1	0	0	0	0	0		-1.5
2	5	0	0	0	0	1	0	0	0	0		-2
	6	0	0	0	0	0	1	0	0	0		0
	7	0	0	0	0	0	0	1	0	0		2
3	8	0	0	0	0	0	0	0	1	0		.5
	9	0	0	0	0	0	0	0	0	1		-.5

At this point, prior utilities have been established and attributes are carried over into the pairs section of the conjoint. In a preferred embodiment, if an attribute has more than five levels (even after removing levels due to constraints or unacceptables) then only the top five ranked levels for that attribute are carried into the pairs section (ties are broken by the predetermined order of the levels). In other embodiments, more or fewer of the top ranked levels may be carried into the pairs section.

For the sake of simplicity to the user, in the preferred embodiment a limit to the number of pairs questions used is set at seven pairs questions (compared to 20 or more for traditional conjoint). As a result the number of levels that are included in the pairs section of the study is limited. This can be accomplished by bringing forward to the pairs section the top five attributes in importance (ties are broken by the predetermined

order of the attributes). Within attributes at most five levels from each (not to exceed a total of 25 levels) will be brought forward. Other embodiments may impose greater or lesser limits to the number of pairs questions. In a preferred embodiment, the number of pairs questions actually presented may vary within the set limit by the individual purchaser, who may choose to stop answering questions and receive recommendations at any time.

Setting Attribute Constraints and Identifying Unacceptables

Constraint specification allows the user to specify constraints on attributes and levels to be included in the product consideration set. For example, a user may set a specific maximum price limit they are willing to pay, or a minimum hard drive capacity which they are willing to accept.

Users will be encouraged to avoid using constraints (their use limits the options available to the search algorithm), but may be allowed to do so anyway. As a result the system will need to be able to adjust accordingly both the levels included in the study and the final scoring algorithm used to search for products. For example, if a user specifies a price constraint of $2,500 for a laptop then several adjustments will need to be made.

First, all levels above $2,500 within the price attribute will need to be removed for the duration of the study. However, the attribute level representing $2,500, or the level immediately greater than $2,500 will need to be retained in the study.

- If the constraint value, $2,500, is a defined level then we simply eliminate all other levels above that value.
- If it is not a defined level, then it can be handled in one of two ways.
  - The first, but less clean method, would be to simply keep the closest level to the constraint level. For example, if a study had levels of $1,750, $2,250, $2,750, $3,250, and $3,750, then the last two levels would be removed and the first three would remain (i.e. we would keep the closest inclusive level to $2,500 or $2,750).
  - The second and preferred, but perhaps more involved method would be to create a new level for the constraint and re-scale the other levels accordingly.

[Note that the preferred definition of numeric levels for conjoint is that they be either equidistant, or in an exponential relationship to each other]

Equidistant (by $500)	Exponential ($100*2^x)	None (not preferred)
$500	$200	$400
$1000	$400	$750
$1500	$800	$1,000
$2000	$1600	$1,600

Again, suppose we had levels of $1,750, $2,250, $2,750, $3,250, and $3,750. Using the second approach eliminates the top three levels and add a level for the constraint value, $2,500, leaving three levels: $1,750, $2,250, $2,500. Since these levels are not equidistant, the intermediate level(s) are re-scaled so that they would be: $1,750, $2,125, and $2,500.

Finally, the scoring algorithm would need to reflect the specification of a constraint in that the system would be instructed not to return any products that exceed the constraint – i.e. no laptops with a price greater than $2,500 (regardless of the other features).

- For some embodiments, hard constraints such as $2,500 can be exceeded by a certain percentage (10% or 20%), at the request of the system administrator, to allow for the fact that users are often flexible in the price they are willing to pay if presented with a superior product.

Another variation on the constraint theme, is the asking of unacceptables. This question type may or may not be used in certain embodiments. In this type of question, users are presented with all of the levels for each attribute and asked to indicate those that are absolutely unacceptable.

- Again, users will be encouraged to limit the number of levels they declare unacceptable.
- At least two levels must remain for each attribute.

- Once unacceptable levels have been specified they are removed from the remainder of the study. The scoring algorithm also needs to be updated to reflect the fact that no products containing an unacceptable level should be returned.

Special consideration needs to be employed with respect to the brand attribute. Since the conjoint is only capable of handling a certain number of levels per attribute, and since the number of brands available in a merchant's product database can be quite large, constraints are specified on the brand attribute to accommodate this.

Three potential approaches to doing this are:

1) List all possible brands and have consumers identify up to X number of brands they will not accept. The remaining brands then get presented again to the user during the ranking questions where the user will be asked to rank their top five.
2) Present users with a list of brands available in the study. The user is asked to place all attributes into one of three categories – 'unacceptable', 'acceptable' and 'most desired' ('acceptable' would be the default and would require no user action). Only 'most desired' brands (up to a Maximum X) would be carried into the conjoint.
3) Present users with up to five drop down boxes so that they can choose their five most preferred brands.

Regardless of method used, the consumer would also be given the option to say that all brands are acceptable. In this case NO brands (i.e. no brand attribute) are carried forward to the conjoint – the purchaser is relatively brand neutral.

There may be several types of attributes that have too many levels to be effectively included in the conjoint – color may be another example. If this attribute is determined to be un-correlated with the remaining attributes, then the attribute may simply be excluded. For example, if a buyer can choose any model and then simply select the color they want, then color does not need to be included as an important decision criterion. However, if the attribute is correlated with others – say some colors cost more or certain models are unavailable in certain colors – then it must be handled in a manner similar to that discussed for "brand" above. All of these decisions can be made by the system administrator and specified in the system configurations.

Conjoint Pairs Section

The conjoint pairs section places users in actual decision-making situations. They are presented with pairs of hypothetical, but realistic, products or services, and then asked to choose which of the pair they prefer, and how strongly they prefer it.

Unlike full profile conjoint techniques that require all product features be evaluated in each product pair, adaptive conjoint presents only a subset of attributes at a time. More than four attributes are rarely shown at once, although five may be shown occasionally. In fact, most begin with several screens comparing products using only two attributes at one time, and then several screens showing three attributes.

The pairs section relies on a calculation procedure based upon multiple regression routines, that is run after each pair. The calculation procedure is designed to accomplish two tasks:

- First, the calculations are used to update the estimated utilities for each level of each attribute. This requires a matrix of the values of the independent variables (the questions) and a vector of the values of the dependent variables (the answers).
- Second, based on the previous pair shown and the respondent's answer (or preference rating), the calculation is used to determine which pairs to show next. Since this is an interactive process, both the utilities for levels and for potential product configurations are estimated repeatedly to select the next question – the one that will provide the most useful information in refining the utility estimations for the individual respondent.

Determining Number of Paris to be shown

The number of pairs to be shown is a function of the number of attributes and levels in the study. Some of the elements required in pairs calculations are detailed below:

- **NATT** = # of Attributes = 3
- **NLA(i)** = # of levels in ith attribute where I = 1 to NATT NLA(1) = 4
  NLA(2) = 3
  NLA(3) = 2
- **NL** = total # of levels in the study 4+3+2 = 9

- NAP = # of available unique products with one level from each attribute

  4*3*2 = 24

- NAP2 = # of available unique products when only two attributes are shown at one time = Σ NLA(i)*NLA(j) = (4*3)+(4*2)+(3*2) = 26

- NPR = # of pairs questions that need to be asked = 3(NL-NATT-1)-NL = 3(9-3-1)-9 = 6

Selecting Pairs to be Shown

In a preferred embodiment, the number of pairs to be shown will be limited to seven pairs in the following format:

Number of Attributes per Pair	Number of Pairs of this kind Shown
2	2
3	4
4	1

Choosing the pairs to be shown

The first pairs question is selected using only the priors data. Only two attributes will be used in the initial pairs questions, and will later expand to three attributes once several questions have been asked and the web visitor accumulates some experience with this type of question. The pairs shown are designed with two purposes in mind – to keep the design balanced (all attributes and levels shown roughly the same number of times) and to present the pairs that will require the most difficult decision from the user and thus give the most information to the software. There are three general steps involved in choosing the pairs to be shown.

1) Identify the set of attributes (2 or three depending on how many will be shown in each pair) that have been shown together the least often. Break a tie randomly.
2) Identify the pairs of levels (one to be shown on each side of the screen) within each of the chosen attributes that have been shown together the least often. Break a tie randomly.

3) From the many possible product combinations that can be generated using the chosen attributes and levels, identify the two that have the most similar total utility value (each being defined as the sum of utilities for the levels making up the product)

Each of these three steps is described in more detail below:

1) Identify the set of attributes (2 or three depending on how many will be shown in each pair) that have been shown together the least often. Break a tie randomly.

To keep track of how often each attribute is used in a question with every other attribute a simple n X n matrix is set up, where n equals NATT:

Attribute	1	2	3
1	0	0	0
2	0	0	0
3	0	0	0

Initially all values are 0. Each element is incremented by one each time the row and column attributes are used together in a question. The attribute combination with the lowest value is chosen. Ties are broken randomly.

The analysis according to the present invention, in order to begin with a balanced design, preorders the attributes for the first NATT pairs. The first pair is always attributes 1 and 2. Thus if two attributes were to be shown at a time, in a study of three attributes, the next two pairs would use attributes 2 and 3 and attributes 3 and 1. Assuming the first pair uses attributes 1 and 2, the matrix above would be updated.

Attribute	1	2	3
1		1	0
2			0
3			

Note: To make the updating process easier, only the top half of the matrix is used. The diagonal or identity array has also been deleted, as it is not relevant here.

Since only a small number of pairs are being used, in the preferred embodiment, it is possible to pre-ordain all of the attributes selected for each of the pairs to ensure as efficient and orthogonal a design as possible. In an alternate embodiment, the selection approach as detailed above may be used.

Possible Pre-Selected Attribute Pairs for Precision Choice

Pair	NATT	1st Attribute	2nd Attribute	3rd Attribute	4th Attribute
1	2	Att. 1	Att. 2		
2	2	Att. 3	Att. 4		
3	3	Att. 1	Att. 3	Att. 5	
4	3	Att. 1	Att. 4	Att. 5	
5	3	Att. 2	Att. 3	Att. 5	
6	3	Att. 2	Att. 4	Att. 5	
7	4	Att. 1	Att. 2	Att. 3	Att. 4

This will ensure that each attribute pairing will be shown together an equal number of times (two times for each combination). Levels and products will still be selected as described.

2) Identify the pairs of levels (one to be shown on each side of the screen) within each of the chosen attributes that have been shown together the least often. Break a tie randomly.

To keep track of how often each level in a selected attribute has been used together with every other level, similar matrices are created for each attribute. Since attribute 3 has only two levels, it is not necessary to create a matrix – both levels will always be used when the attribute is selected.

Attribute 1

Level	1	2	3	4
1		0	0	0

2			0	0
3				0
4				

Attribute 2

Level	5	6	7
5		0	0
6			0
7			

Levels are chosen randomly if there is more than one combination that has been used the fewest number of times. In the example, the first pair chosen uses Attributes 1 and 2. The selected levels were 1 and 3 and 6 and 7 (1st and 3rd from Attribute 1 and 2nd and 3rd from Attribute 2). The matrices above are then updated accordingly. For example, if the first question pairs levels 1 and 3 in Attribute 1 and levels 6 and 7 in Attribute 2, then the elements (1,3) and (6,7) are each increased by 1.

Attribute 1

Level	1	2	3	4
1		0	1	0
2			0	0
3				0
4				

Attribute 2

Level	5	6	7
5		0	0
6			1

7			

3) From the many possible product combinations that can be generated using the chosen attributes and levels, identify the two that have the most similar total utility value (each being defined as the sum of utilities for the levels making up the product)

The priors utilities are used to estimate the total utility for each possible product (a combination of features that could be shown). This requires one or more tables of the total utilities of each product and one or more matrices representing the differences in the total utilities between all possible pairs of products.

The pair of products chosen for a question is the pair of products that have the most similar total utility. Break a tie randomly. Randomly assign one product to the left-hand side of the question screen and the other to the right hand side. The purpose of this step is to present to the web visitor a choice between two products thought from his/her earlier responses to be valued closely. This is a credible question to ask the web visitor and the answer provides meaningful information in the interactive process of calculating the total utility for each product.

For 3 attributes with a total of nine levels, as in the example, there are 24 possible products (4*3*2) when three attributes are shown at a time and 26 possible products (4*3+4*2+3*2) when only two are shown. Several tables as shown below can represent these permutations and the utilities for each.

		Attribute			Product Utility			Total
Attributes	**Product**	**1**	**2**	**3**	**Prior1**	**Prior2**	**Prior3**	**Priors**
1&2	1	1	1		1.5	-2		-0.5
	2	2	1		0.5	-2		-1.5
	3	3	1		-0.5	-2		-2.5
	4	4	1		-1.5	-2		-3.5
	5	1	2		1.5	0		1.5
	6	2	2		0.5	0		0.5
	7	3	2		-0.5	0		-0.5
	8	4	2		-1.5	0		-1.5

	9	1	3		1.5	2		3.5
	10	2	3		0.5	2		2.5
	11	3	3		-0.5	2		1.5
	12	4	3		-1.5	2		0.5

		Attribute			Product Utility			Total
Attributes	Product	1	2	3	Prior1	Prior2	Prior3	Priors
1&3	13	1		1	1.5		0.5	2
	14	2		1	0.5		0.5	1
	15	3		1	-0.5		0.5	0
	16	4		1	-1.5		0.5	-1
	17	1		2	1.5		-0.5	1
	18	2		2	0.5		-0.5	0
	19	3		2	-0.5		-0.5	-1
	20	4		2	-1.5		-0.5	-2

		Attribute			Product Utility			Total
Attributes	Product	1	2	3	Prior1	Prior2	Prior3	Priors
2&3	21		1	1		-2	0.5	-1.5
	22		2	1		0	0.5	0.5
	23		3	1		2	0.5	2.5
	24		1	2		-2	-0.5	-2.5
	25		2	2		0	-0.5	-0.5
	26		3	2		2	-0.5	1.5

		Attribute			Product Utility			Total
Attributes	Product	1	2	3	Prior1	Prior2	Prior3	Priors
1&2&3	1	1	1	1	1.5	-2	0.5	0
	2	2	1	1	0.5	-2	0.5	-1
	3	3	1	1	-0.5	-2	0.5	-2
	4	4	1	1	-1.5	-2	0.5	-3
	5	1	2	1	1.5	0	0.5	2
	6	2	2	1	0.5	0	0.5	1
	7	3	2	1	-0.5	0	0.5	0
	8	4	2	1	-1.5	0	0.5	-1

	9	1	3	1	1.5	2	0.5	4
	10	2	3	1	0.5	2	0.5	3
	11	3	3	1	-0.5	2	0.5	2
	12	4	3	1	-1.5	2	0.5	1
	13	1	1	2	1.5	-2	-0.5	-1
	14	2	1	2	0.5	-2	-0.5	-2
	15	3	1	2	-0.5	-2	-0.5	-3
	16	4	1	2	-1.5	-2	-0.5	-4
	17	1	2	2	1.5	0	-0.5	1
	18	2	2	2	0.5	0	-0.5	0
	19	3	2	2	-0.5	0	-0.5	-1
	20	4	2	2	-1.5	0	-0.5	-2
	21	1	3	2	1.5	2	-0.5	3
	22	2	3	2	0.5	2	-0.5	2
	23	3	3	2	-0.5	2	-0.5	1
	24	4	3	2	-1.5	2	-0.5	0

Once all of the possible products have been outlined and their total utilities calculated based on the priors, a matrix of the absolute values of the differences between total product utilities can be constructed. Only half of each of the matrices shown below needs to be filled in.

Looking at the following tables of product pairs choose the appropriate table based upon the attributes and levels shown. Within the subset of products possible from those levels and attributes, the pairs of products with the most similar utility (i.e. zero values except for the principal diagonal) must be identified in the matrix so that one of them can be selected as the first question to be asked.

In general, these are the rules to use:

1) Calculate the difference in total priors utilities between each possible product combination.
2) Find the minimum value in the matrix by examining all the elements. Identify the two products associated with that minimum value. Identify any other product pair that has the identical minimum value. Record the products associated with those elements.

3) If the minimum value occurs only once, the two products associated with that value are the products to use in the first pairs question.

4) If the minimum value occurs more than once (this will certainty be the case for the first question in every experiment), then randomly select one of the occurrences; the two products associated with that selection are used in the first pairs question. In an alternate embodiment, the first occurrence may be selected instead of by random selection.

Attribute	Product	1	2	3	4	5	6	7	8	9	10	11	12
1&2	1	0	1	2	3	2	1	0	1	4	3	2	1
	2		0	1	2	3	2	1	0	5	4	3	2
	3			0	1	4	3	2	1	6	5	4	3
	4				0	5	4	3	2	7	6	5	4
	5					0	1	2	3	2	1	0	1
	6						0	1	2	3	2	1	0
	7							0	1	4	3	2	1
	8								0	5	4	3	2
	9									0	1	2	3
	10										0	1	2
	11											0	1
	12												0

Attribute	Product	13	14	15	16	17	18	19	20				
1&3	13	0	1	2	3	1	2	3	4				
	14		0	1	2	0	1	2	3				
	15			0	1	1	0	1	2				
	16				0	2	1	0	1				
	17					0	1	2	3				
	18						0	1	2				
	19							0	1				
	20								0				

Attribute	Product	21	22	23	24	25	26						

2&3	21	0	2	4	1	1	3						
	22		0	2	3	1	1						
	23			0	5	3	1						
	24				0	2	4						
	25					0	2						
	26						0						

Attribute	Product	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1&2&3	1	0	1	2	3	2	1	0	1	4	3	2	1	1	2	3	4	1	0	1	2	3	2	1	0
	2		0	1	2	3	2	1	0	5	4	3	2	0	1	2	3	2	1	0	1	4	3	2	1
	3			0	1	4	3	2	1	6	5	4	3	1	0	1	2	3	2	1	0	5	4	3	2
	4				0	5	4	3	2	7	6	5	4	2	1	0	1	4	3	2	1	6	5	4	3
	5					0	1	2	3	2	1	0	1	3	4	5	6	1	2	3	4	1	0	1	2
	6						0	1	2	3	2	1	0	2	3	4	5	0	1	2	3	2	1	0	1
	7							0	1	4	3	2	1	1	2	3	4	1	0	1	2	3	2	1	0
	8								0	5	4	3	2	0	1	2	3	2	1	0	1	4	3	2	1
	9									0	1	2	3	5	6	7	8	3	4	5	6	1	2	3	4
	10										0	1	2	4	5	6	7	2	3	4	5	0	1	2	3
	11											0	1	3	4	5	6	1	2	3	4	1	0	1	2
	12												0	2	3	4	5	0	1	2	3	2	1	0	1
	13													0	1	2	3	2	1	0	1	4	3	2	1
	14														0	1	2	3	2	1	0	5	4	3	2
	15															0	1	4	3	2	1	6	5	4	3
	16																0	5	4	3	2	7	6	5	4
	17																	0	1	2	3	2	1	0	1
	18																		0	1	2	3	2	1	0
	19																			0	1	4	3	2	1
	20																				0	5	4	3	2
	21																					0	1	2	3
	22																						0	1	2
	23																							0	1
	24																								0

As an alternate approach, a slightly modified procedure may be used to ensure that the question asked is not a choice between two products, either both very desirable or very undesirable. Instead of breaking the tie randomly, select the pair of products whose total utility is the median of all the product sets involved in the tie. Then select randomly if there is another tie.

While the above represent the entire matrices of possible product combinations, in reality only a subset of these matrices needs to be considered each time – only the subset of product combinations that can be created out of the selected levels within the chosen attributes. For a pair of NATT attributes on each side of the screen, with two levels of each attribute shown, the number of possible product configurations that can be created from those levels is equal to 2^{NATT}. So for a two-attribute pair using attributes 1&2 in the example above, only 2^2 or 4 of the 12 possible products will need to be compared.

Calculating Equal Weighted Utilities

As mentioned earlier, the first question in the example pairs contains levels 1 and 3 in attribute 1 with levels 6 and 7 in attribute 2. This results in four possible product combinations from our product matrix of 26 possible products:

	Attribute			Product Utility			Total
Product	1	2	3	Prior1	Prior2	Prior3	Priors
5	1	2		1.5	0		1.5
7	3	2		-0.5	0		-0.5
9	1	3		1.5	2		3.5
11	3	3		-0.5	2		1.5

Choosing the products with the least difference in total utility from the priors would result in product 5 and product 11 being shown together. Choosing randomly, product 11 is shown on the left side and product 5 is shown on the right.

Once the first pair of products to be shown has been chosen the pairs information is appended as an additional row to the priors matrix. Attribute levels shown on the left side of the screen (levels 3 and 7) are coded as –1 while levels shown on the right (levels 1 and 6) are coded as 1. Levels that are not shown are coded as zero.

The respondent's answer, given on a nine-point scale is re-centered at zero by subtracting five from the answer given. Assuming the respondent answered 4 to the first question, the updated matrix would be as follows:

Priors Utility Matrix

	Independent X variables (levels)										
Level	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**Dependent Y**
1	1	0	0	0	0	0	0	0	0		1.5
2	0	1	0	0	0	0	0	0	0		.5
3	0	0	1	0	0	0	0	0	0		-.5
4	0	0	0	1	0	0	0	0	0		-1.5
5	0	0	0	0	1	0	0	0	0		-2
6	0	0	0	0	0	1	0	0	0		0
7	0	0	0	0	0	0	1	0	0		2
8	0	0	0	0	0	0	0	1	0		.5
9	0	0	0	0	0	0	0	0	1		-.5
Pair1	1	0	-1	0	0	1	-1	0	0		-1

A multiple regression is then run across the nine independent variables (levels) and one dependent variable (utilities/answers) across all 10 cases in the dataset (rows). The regression, will produce nine coefficients for each of our independent variables. These nine values are similar to the priors utilities (first nine rows in the Dependent Y column) in the matrix above. These new utility values will replace the priors values in the product tables and in the calculation of total product utilities. However, the original prior values will be retained for the regression routines.

Recalculate utilities in the Regression Module **After first Pair**

After the first pair in the example, the regression routine produces the following coefficients.

X1	1.3
X2	.5
X3	-.3
X4	-1.5
X5	-2
X6	-.2
X7	2.2
X8	.5
X9	-.5

Note that the coefficients for the 1st, 3rd, 6th and 7th levels have changed because the first pairs question had additional preference information about these four levels. The remaining coefficients remain unchanged.

To continue with this example, the answers to the next five pairs questions presented will be used. After the answer to each pairs question is incorporated:

a) New utilities for each level are estimated

b) New total utilities for each unique product concept are estimated

c) New differences in total utility between possible sets of pairs are estimated

After the product matrices have been updated, the next pair is chosen according to the same rules:

1) Identify the set of attributes (2 or three depending on how many will be shown in each pair) that have been shown together the least often. Break a tie randomly.
2) Identify the pairs of levels (one to be shown on each side of the screen) within each of the chosen attributes that have been shown together the least often. Break a tie randomly.
3) From the many possible product combinations that can be generated using the chosen attributes and levels, identify the two that have the most similar total utility value (each being defined as the sum of utilities for the levels making up the product). This will be based on the new utility values in the updated matrix.

This process will be repeated after every pair until all of the required 3*(NATT-NL-1)-NL pairs have been asked. In the example, six pairs questions needed to be asked, at which point our priors matrix will look like the following:

	Independent X variables (levels)										
Level	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**Dependent Y**
1	1	0	0	0	0	0	0	0	0		1.5
2	0	1	0	0	0	0	0	0	0		.5
3	0	0	1	0	0	0	0	0	0		-.5
4	0	0	0	1	0	0	0	0	0		-1.5
5	0	0	0	0	1	0	0	0	0		-2
6	0	0	0	0	0	1	0	0	0		0
7	0	0	0	0	0	0	1	0	0		2
8	0	0	0	0	0	0	0	1	0		.5
9	0	0	0	0	0	0	0	0	1		-.5
Pair1	1	0	-1	0	0	1	-1	0	0		-1
Pair2	-1	1	0	0	0	0	0	1	-1		0
Pair3	0	0	0	0	-1	0	1	-1	1		2
Pair4	-1	0	0	1	0	0	0	1	-1		-3
Pair5	0	0	-1	1	-1	1	0	1	-1		-1
Pair6	0	1	-1	0	1	0	-1	1	-1		-3

Pairs 5 and 6 were products composed of three attributes each instead of two (represented by six rather than four non-zero values). The model did not require this

change. However, three attribute questions are able to provide more information to the regression equation and calculation of utilities.

Once all of the pairs have been asked and the matrix updated, a final regression is run to calculate what are called the "final equal weight utilities". These values are a standard output of conjoint and are used to help choose the products shown in the calibration questions (explained in detail in that section).

For the example, the final equal weighted utilities produced by the regression are:

X1	1.415441
X2	0.370098
X3	0.316176
X4	-2.101716
X5	-1.471814
X6	-0.346814
X7	1.818627
X8	0.352941
X9	-0.352941

Prohibition of Pairs

Note, in addition to the process described above for choosing product pairs, sometimes it is necessary to prohibit certain pairs of attributes/levels from being shown together. Specifically, in some studies there are combinations of levels that should not be shown together – i.e. they will not make sense. Examples may include bill presentment without bill payment for a banking product or a bagel and high fat for takeout breakfast foods. (Bill payment functionality is usually a prerequisite to bill presentment, and bagels are by definition non-fat.)

Calculating Pairs Only Utilities

Another critical component of conjoint is the calculation of the pairs utilities, isolated from the priors. This enables the understanding of how the products presented in the pairs related directly to the respondent's preferences. This is important so that

the priors and pairs can be properly weighted to come up with an accurate overall utility estimation for the respondent.

Calculation of the pairs comes from a multiple regression utilizing only the pairs piece of the matrix shown previously (with 9 levels as independent variables and Y as our dependent variable). An identity matrix with the dependent variables set at zero is placed below the pairs values to stabilize the regression, resulting in the following matrix:

	Independent X variables (levels)										
Level	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**Dependent Y**
Pair1	1	0	-1	0	0	1	-1	0	0		-1
Pair2	-1	1	0	0	0	0	0	1	-1		0
Pair3	0	0	0	0	-1	0	1	-1	1		2
Pair4	-1	0	0	1	0	0	0	1	-1		-3
Pair5	0	0	-1	1	-1	1	0	1	-1		-1
Pair6	0	1	-1	0	1	0	-1	1	-1		-3
Identity1	1	0	0	0	0	0	0	0	0		0
Identity 2	0	1	0	0	0	0	0	0	0		0
Identity 3	0	0	1	0	0	0	0	0	0		0
Identity 4	0	0	0	1	0	0	0	0	0		0
Identity 5	0	0	0	0	1	0	0	0	0		0
Identity	0	0	0	0	0	1	0	0	0		0

6											
Identity 7	0	0	0	0	0	0	1	0	0		0
Identity 8	0	0	0	0	0	0	0	1	0		0
Identity 9	0	0	0	0	0	0	0	0	1		0

Performing a multiple regression on the above matrix (and setting the intercept to zero) will yield nine coefficients which are our pairs utilities for the nine levels in the study. The regression results yield the following coefficients as estimates of the utilities contributed by the pairs. These are the pairs only utilities.

X1	0.151961
X2	0.218954
X3	0.460784
X4	-0.83170
X5	-0.71732
X6	0.116013
X7	0.601307
X8	-0.43137
X9	0.431373

These are still RELATIVE NOT ABSOLUTE utilities because the response scale for the questions was arbitrary.

Assigning Utilities to omitted levels

If the number of levels is large some levels are likely not to be included in any paired questions. Extreme undesirables are the most likely to be omitted because they are less likely to add significant value to the understanding of the decision making process. With a very large number of levels some levels with non-extreme utilities may

also be omitted. Levels with missing values have all zeros in the column for the level in the pairs section of the independent variable matrix.

Many levels may be omitted as we will only be bringing the 5 most important attributes and three most important levels of each into the pairs section. Some levels, i.e. brands may even be omitted from the ranking sections.

After the last pairs question has been asked, answered and the utilities re-estimated, values must be assigned to any missing values before weighting them with the prior utilities.

Rules:

1) The rank order of desirability is known for all levels from the priors stage. If two levels had the same prior utility, they would have the same rank order. If a level with a missing pairs utility value has a prior rank value between the rank values for levels for which pairs utilities have been estimated, the missing value is estimated by interpolation.
   - For example: Suppose Level 1 has a missing value (never used in a pairs question) and its desirability is 3. Suppose also that Level 2 has a desirability of 4 and an estimated utility coefficient of 1.55 while Level 3 has a desirability of 2 and an estimated utility coefficient of 1.75.
   - The desirability of Level 1 is midway between Level 2 and Level 3. So the estimated utility for Level 1 is set midway between the utilities of Level 2 and Level 3. Midway is 1.65. Thus Level 1 is assigned a utility coefficient of 1.65.
   - If another level also has the same desirability as one of the levels used in interpolation, such as Level 3, then the utility used in interpolation is the average between the additional level and Level 3.
2) If any level with missing values has a desirability more extreme than the desirabilites for levels for which utilities have been estimated, then the utility for the level with the missing value is set, as appropriate, at the maximum or minimum utility estimated for any level for which a utility estimate has been made.
   - For example: Suppose Level 5 has a missing value (never used in a pairs question) and its desirability is 6. Suppose that Level 6 has a desirability of 5,

an estimated utility coefficient of 1.55 and was the lowest ranked level included in the pairs section.

- In this case, since we cannot extrapolate beyond the value of Level 6 (the extreme minimum value), Level 5 would be given the same utility value of 1.55.

As another example, assume we had a study with 3 attributes and 9 levels (different from our current example).

If Prior utilities were as in the table below, and levels 2,4,8 and 9 were not included in the pairs, the pairs utilities for those levels could be estimated. This procedure can be undertaken even if a whole attribute was excluded from the conjoint pairs section – Attribute 3 below.

Attribute	Level	Prior Utility	Observed Pairs Utility	Estimated Pairs Utility
1	1	1.5	.980	.980
	2	.5	–	.419*
	3	-.5	-.186	-.186
	4	-1.5	–	-.508**
2	5	1	.542	.542
	6	0	.296	.296
	7	-1	-.508	-.508
3	8	.5	–	.419*
	9	-.5	–	-.186***

*Pairs utility estimate for levels 2 and 8 is based on an interpolation between the closest levels for which there are pairs values. Levels 2 and 8 both had a prior value of .5. The nearest values are 0 (Level 6) and 1 (Level 5). Since .5 is halfway between 0 and 1, Levels 2 and 8 are assigned the mid-point of .542 and .296.

**Since the prior value for Level 4 is lower than any other prior value it is assigned the pairs utility equal to that of the level with the minimum prior value for which a pairs value was calculated. Thus it is assigned a value of -.508 from Level 7.

***Since Level 9 has the same prior value as Level 3 it is assigned the same pairs utility.

Assigning utilities to levels omitted through unacceptables

There may also be levels that were not included in either the priors or the pairs section of the conjoint. These would include levels from attributes with many levels such as brand – levels beyond those ranked in the top five (not included in priors rankings or pairs) and would also include levels eliminated through unacceptables. These adjustments will be made based upon the combined optimal weight utilities after the calibration concepts, rather than based on priors utilities as above. This is described in more detail below.

Calibration Concepts

After the pairs section is complete, the application will have computed two sets of preference data (i.e., utilities) for a user – a set of priors utilities based on the original ranking and importance questions and a set of pairs utilities based on the product pairs questions. The next step, in a preferred embodiment, is to ask calibration questions which are used to adjust the respondent's utilities. Other embodiments may forego the calibrating step.

The user will be asked a series of four questions that help the software finish its evaluation. In each question, they will be shown a hypothetical product and will be asked to rate how likely they would be to choose, use, or purchase (terminology will depend on the type of product/service being analyzed) this product on a scale of 0 to 100%. Users are instructed to interpret 0% to mean they would absolutely not use the product and 100% to mean that they would definitely use the product if it were available.

How the Calibration Concepts Work

Each question in the calibration section asks the respondent to evaluate a product, which is described by up to five attributes – the five attributes determined to be the most important to the user based on the analysis of equal weight utilities. Since our example only contains three attributes only three attributes are used.

In the preferred embodiment, the first product contains the worst level of each attribute, and asks the consumer to rate how likely they would be to buy, from 0% to 100% likely. The second question offers the best level of each attribute and again asks the consumer to rate how likely they would be to buy. The third and subsequent profiles

choose a combination of attribute levels that should fall in the middle-range of attractiveness.

The answers to the calibrations are used for three purposes:

1) To determine how to weight the priors utilities versus the pairs utilities to get final optimal weighted respondent utilities,

- To do this, responses to the calibration questions are compared to what would be predicted by the prior and pairs utilities

2) To determine how to scale the utilities for that particular respondent,

- Utility values are scaled based upon the range between the answers to the first and second (worst and best) calibration concepts.
- If the range of answers to the calibration questions is narrow, then the consumer is not really "in the market" and their utilities will be scaled smaller (i.e. a narrow range). Conversely, if the range between responses is high, then they are more discerning and the utilities are scaled larger. Note that this process is especially important when looking at consumers in the aggregate, or when merging profiles between users, to determine how respondents should be weighted. By scaling the utilities according to their level of discernment between the worst and best products, aggregating customers becomes a simple process of averaging their utilities.

3) To determine the internal consistency of the respondent between the priors and the pairs

- The goodness of fit is measured for the regression. When examining aggregate data, it is often useful to eliminate respondents with a poor internal consistency of responses.

Select first two calibration questions

The first two questions are selected using the estimated equal weight utilities (prior plus pairs) after any missing values for level pairs utilities have been inserted and the utilities restated. The first calibration question will be the full profile product concept with the least desirability (lowest total utility) based on the web visitors'

answers. The second question will be the full profile product concept with the most desirability.

The utilities of the levels in the example are reproduced below.

Level	Prior utilities	Pairs utilities	Prior plus pairs equal weight utilities
x1	1.5	0.151961	1.415441
x2	.5	0.218954	0.370098
x3	-.5	0.460784	0.316176
x4	-1.5	-0.83170	-2.010172
x5	-2.0	-0.71732	-1.47181
x6	0.0	0.116013	-0.34681
x7	2.0	0.601307	1.818627
x8	.5	-0.43137	0.352941
x9	-.5	0.431373	0.35294

These values can be used to calculate the total utilities for each product combination by taking the sum of the utilities for each level in the product, as below:

Product	Level From Attribute 1	Level From Attribute 2	Level From Attribute 3	Total Priors Utility	Total Pairs Utility	Total prior plus pairs equal weight utilities
1	1	5	8	0	-0.996729	0.296572
2	2	5	8	-1	-0.929736	-0.739069
3	3	5	8	-2	-0.687906	-0.802693
4	4	5	8	-3	-1.98039	-3.129041
5	1	6	8	2	-0.163396	1.421572
6	2	6	8	1	-0.096403	0.376229
7	3	6	8	0	0.145427	0.322307
8	4	6	8	-1	-	2.004041

					1.147057	
9	1	7	8	4	0.321898	3.587009
10	2	7	8	3	0.388891	2.541666
11	3	7	8	2	0.630721	2.487744
12	4	7	8	1	-0.661763	0.161396
13	1	5	9	-1	-0.133986	0.296572
14	2	5	9	-2	-0.066993	-0.739069
15	3	5	9	-3	0.174837	-0.802693
16	4	5	9	-4	-1.117647	-3.129041
17	1	6	9	1	0.699347	1.421572
18	2	6	9	0	0.76634	0.376229
19	3	6	9	-1	1.00817	0.322307
20	4	6	9	-2	-0.284314	2.004041
21	1	7	9	3	1.184641	3.587009
22	2	7	9	2	1.251634	2.541666
23	3	7	9	1	1.493464	2.487744
24	4	7	9	0	0.20098	0.161396

Note that in the table above:

- Levels 8 & 9 have the same utility so the total equal weight utilities for products 13-24 are identical to the utilities for products 1-12.

- Products 4 and 16 (pick one randomly) have the lowest total equal weight utility. One of these will be used as the first calibration question. Product 4 was chosen in the example.
- Products 9 and 21 (pick one randomly) have the highest total equal weight utility. One of these will be used as the second calibration question. Product 21 was chosen in the example.

Select remaining calibration questions

Subsequent calibration questions will use the products which have the largest difference in total utility between the pairs only and the priors only estimates. This is to be sure that the questions asked are useful – i.e. to get information to clear up discrepancies. Ties are broken randomly.

Product	Level From Attribute 1	Level From Attribute 2	Level From Attribute 3	Total Priors Utility	Total Pairs Utility	Difference between priors and pairs (AV)
1	1	5	8	0	-0.996729	0.996729
2	2	5	8	-1	-0.929736	0.070264
3	3	5	8	-2	-0.687906	1.312094
4	*4*	*5*	*8*	*-3*	*-1.98039*	*1.01961*
5	1	6	8	2	-0.163396	2.163396
6	2	6	8	1	-0.096403	1.096403
7	3	6	8	0	0.145427	0.145427
8	4	6	8	-1	-1.147057	0.147057
9	1	7	8	4	0.321898	3.678102
10	2	7	8	3	0.388891	2.611109

11	3	7	8	2	0.630721	1.369279
12	4	7	8	1	-0.661763	1.661763
13	1	5	9	-1	-0.133986	0.866014
14	2	5	9	-2	-0.066993	1.933007
15	3	5	9	-3	0.174837	3.174837
16	4	5	9	-4	-1.117647	2.882353
17	1	6	9	1	0.699347	0.300653
18	2	6	9	0	0.76634	0.76634
19	3	6	9	-1	1.00817	2.00817
20	4	6	9	-2	-0.284314	1.715686
21	*1*	*7*	*9*	*3*	*1.184641*	*1.815359*
22	2	7	9	2	1.251634	0.748366
23	3	7	9	1	1.493464	0.493464
24	4	7	9	0	0.20098	0.20098

Products 4 and 21 have been shown. Of the remaining products, product 9 has the greatest difference in total utility between the priors and the pairs. It will be the third calibration concept. The fourth concept will be the product with the next highest difference, product 15.

Ask calibration questions and receive answers

The answers to the calibration questions are on a scale from 0 to 100. First answers are adjusted, if necessary, so that the minimum answer is 5 and the maximum answer is 95. This is not re-scaling. Simply raise any number below 5 to 5 and lower any number above 95 to 95. This is done to eliminate extreme values.

Next, the answers are adjusted again so that they equal the logit value of the respondent's answer, using this formula:

Adjusted answer = Ln [answer/(100-answer)]

Thus, if the answer is 10, the adjusted answer is -2.197.

Staying with the example, the adjusted answers are:

Answer	Adjusted answer
10	-2.197
80	1.386
40	-.405
5	-2.944

Calculate the weights for the prior and pairs utilities

To calculate the optimally weighted utilities (the "final utilities"), the software must first calculate the weights for the priors and pairs, respectively. To accomplish this, the application will first calculate user utilities for each product concept that was included in the calibration concept section of the study.

The weights are based on a regression in which the adjusted answers to the calibration questions calculated above are used as the dependent variables, while the independent variables are the values of the total utilities of the product concept asked for 1) the pairs only utilities, and 2) the priors only utilities.

For the example:

Product	Prior total utility	Pairs total utility	Adjusted answer
4	-3	-1.980	-2.197
21	3	1.185	1.386
9	4	0.322	-0.405
16	-4	-1.118	-2.944

This regression can be computed in the same manner as the previous regressions. Note: This regression equation requires that a constant term, or intercept, also be calculated. So the matrix of independent variables has a third column - a vector

of 1's - and the regression module returns three coefficients – the intercept value, the priors coefficient and the pairs coefficient.

However, since there are only two independent variables and we wish to have an intercept, the regression can also be calculated using the formulae below:

$$m1 = \frac{((\Sigma x_1 y^* \Sigma x_2^2)-(\Sigma x_2 y^* \Sigma x_1 x_2))}{(\Sigma x_1^{2*} \Sigma x_2^2)-(\Sigma x_1 x_2)^2}$$

$$m2 = \frac{((\Sigma x_2 y^* \Sigma x_1^2)-(\Sigma x_1 y^* \Sigma x_1 x_2))}{(\Sigma x_1^{2*} \Sigma x_2^2)-(\Sigma x_1 x_2)^2}$$

$$b = \text{MeanY} - (m1^*\text{MeanX}_1) - (m2^*\text{MeanX}_2)$$

Where:

X_{1i} = the ith observation of the priors, X_{2i} = the ith observation of the pairs, and Y_i = the ith observation of the adjusted logit answers.

MeanX_1 equals the mean of the X_1 values, MeanX_2 equals the mean of the X_2 values, and MeanY equals the mean of the Y values.

$x_1 = X_1 - \text{MeanX}_1$, $x_2 = X_2 - \text{MeanX}_2$, and $y = Y - \text{MeanY}$

m1 equals the coefficient for the priors weight, m2 equals the coefficient for the pairs weight, and b equals the intercept.

For the example, the following values are returned from the regression equation:

- coefficient for priors total utility = 0.172
- coefficient for pairs total utility = .805
- intercept = -0.720

Calculation of Final Weighted Utilities

The final utility for each level = prior utility * prior weight + pairs utility * pairs weight + (intercept / NATT)

Example for level 1:

The new utility = 1.500 * 0.172 + .152 * .805 + (-0.720 / 3) = 0.141

The final utilities, in the table below, are now used to calculate the total utility of each product concept in the database to determine which products to present to the web visitor upon search.

X1	0.141

X2	0.023
X3	0.045
X4	-1.168
X5	-1.163
X6	-0.147
X7	0.589
X8	-0.501
X9	0.021

Managing Reversals

During the calculation of the final utilities, in a preferred embodiment, the system must also be able to appropriately deal with reversals. A reversal is where a respondent has explicitly stated that one attribute level is preferred over another during the ranking section, but as a result of the pairs the final utility has the preference reversed. Reversals are often accurate, and thus are allowed to occur in traditional conjoint studies (where the user never sees their results). However, reversals are often difficult for the user to understand, and should therefore be avoided (in the preferred embodiment the user sees her own results). In this case the two utilities should be set equal to each other at the average of their values.

Calculation of R2 on the calibration regression

The system must be capable of calculating R^2 (R squared), which is a mathematical value indicating the goodness of fit between a respondent's answers – in short, how good of a respondent the person was. If this value is low, the results will often be discarded.

The R^2 comes from the regression performed on the calibration responses an can be calculated using the following formula:

$$R^2 = 1-(\Sigma(\text{EstimatedY}-Y)^2 / \Sigma(Y-\text{MeanY})^2)$$

Where:

$$\text{EstimatedY}_i = m1*X_{1i} + m2*X_{2i} + b$$

Accounting for low R-sq respondents

The system must be capable of excluding respondents whose internal consistency of answers, represented by the R-squared, does not meet a certain threshold. The study manager is able to set this threshold. It is usually recommended that the cutoff be set so that approximately 10-15% of the respondents are eliminated – a level of around 0.3 is often sufficient.

It can be handled similarly for the Preferences Database created. Each record will need to be appended with an R-squared value so that it can be filtered when later analyses are run.

During usage of an application according to the present invention – i.e. while being used by a shopper – a user cannot be eliminated per se during the interview. Nonetheless the R-squared value is calculated and databased so that the respondent may be excluded from aggregate level analysis at the discretion of the system administrator.

Calculation of Attribute Relative Importance

One of the main outputs of a conjoint study is an understanding of attribute relative importance – a percentage breakdown of the impact each attribute exerts on the purchase decision/product evaluation process. To calculate relative importance, the system first needs a set of standardized final scaled utilities.

Calculation of Final Scaled Utilities

The system must be capable of calculating the Final Scaled Utilities, which are the final optimal weighted utilities scaled for each attribute so that the lowest level of each attribute has a value of zero and the sum of all utility values is equal to NATT*100.

The calculation of final scaled utilities requires several steps:

- First, the minimum final optimal weight utility value from the levels in each attribute is calculated. For example, in the table below, Attribute 1 has four levels with corresponding utility values of 0.141, 0.023, 0.045, and –1.168. So the minimum or lowest value is –1.168.
- Utilities are then adjusted to be equal to the final optimal weight utility for the level minus the minimum value for the attribute. This will scale the lowest level in each attribute to a utility of zero.

- Next, the sum of all of these new Adjusted Values across all attributes is calculated and divided by the number of attributes. This is done to standardize the values across attributes and ensure that the sum of the utilities is equal to the number of attributes. In our example the sum of 7.004 is divided by NATT (three) to yield an average utility/attribute value of 2.335
- The Scaled Adjusted Value for each level is then calculated by dividing each of the adjusted values by this new value, the average utility/attribute of 2.335
- Finally, these last values are multiplied by 100 and rounded to the nearest whole integer to produce the values in the table below.

Attribute	Level	Final Optimal Weight Utilities	Attribute Min. Value	Adjusted Value (Final–Min.)	Scaled Adjusted Value	Final Scaled Utilities
1	1	0.141	-1.168	1.310	0.561	56
	2	0.023	-1.168	1.191	0.510	51
	3	0.045	-1.168	1.213	0.520	52
	4	-1.168	-1.168	0.000	0.000	0
2	5	-1.163	-1.163	0.000	0.000	0
	6	-0.147	-1.163	1.016	0.435	44
	7	0.589	-1.163	1.752	0.750	75
3	8	-0.501	-0.501	0.000	0.000	0
	9	0.021	-0.501	0.522	0.224	22

Dealing with Omitted Levels

Remember that levels excluded from the conjoint due to unacceptables or other reasons need to be accounted for at this stage. Unacceptables are given a final scaled utility value of –200, while other missing levels are given the value equal to the lowest scoring value for the attribute in question (zero).

Relative Importance

The final scaled utility data, individual scores for each attribute level, is used to calculate the relative weight of each product attribute to the consumer's purchasing

decision. The relative importance of each attribute is based on an analysis of the spread between the highest and lowest scaled utility levels for each attribute.

To calculate relative importance we calculate the absolute strength or value of each attribute, which is equal to the value of the highest level minus the value of the lowest level within the attribute. Since the lowest level has been fixed at zero, effectively the value or strength of the attribute is represented by the utility of its best level. Relative importance of each attribute is then calculated by the value for the attribute divided by the sum of the values across all attributes.

Attribute	Attribute Value (Highest-lowest level)	Percentage	Percentage
1	56	=56/153	36.6%
2	75	=75/153	49.0%
3	22	=22/153	14.4%
Total	**153**	**100%**	**100%**

Attributes then need to be re-sorted in order of their importance for presentation to the user.

Presentation of data to the end-user

This information can be converted into a relative importance chart such as seen in FIG. 2, which shows the impact of each attribute or feature on the respondent's decision making process. In this example for a laptop, processor speed is the most important determinant for this consumer when choosing a laptop, followed by manufacturer brand and by price. Battery life is the least important of the features tested. Each value in the chart above will be rounded to the nearest whole decimal place.

Modify Personal Profile

Consumers can adjust their preference profile to exert some direct control over the process. For example, if after viewing the chart, suppose the consumer felt that processor speed really is not that important to him or her. In this case, he would be able to click on the graph, and one attribute at a time, manually adjust an attribute's importance. The software would automatically adjust the remaining attributes accordingly so that the total remained 100%. So, if the consumer ratcheted down the

importance of processor speed to a 20% weight, the remaining 10% would be proportionally distributed across the other attributes.

The new totals would look as follows:

- Processor Speed 20%
- Brand 22.9%
- Price 22.9%
- RAM 17.1%
- Hard Drive Capacity 11.4%
- Battery Life 5.7%

The user can adjust one attribute at a time. Each time an attribute is adjusted, the relative importance values will be recalculated according to the following procedures:

- The adjusted attribute assumes the new value given by the user. [Note, we may choose to set bands on the extent to which a user can adjust a value, say +/-25%.]
- The remaining values are re-scaled up or down in proportion to their relative importance – more important attributes would gain or lose proportionately more or less of the difference.
- The formula (it works even if multiple attributes, up to NATT-1, are adjusted at once) is:

The new relative importance for Attribute n equals

Attribute n = (1-Σ(new values of adjusted attributes)) / (1-Σ(original values of adjusted attributes))*original value of Attribute n

In the example, suppose the user decided to change the importance of Attribute 1, brand, to 30%.

Attribute	Attribute Value (Highest-lowest level)	Percentage	Percentage	Adjusted Percentage
1	56	=56/153	36.6%	30.0%
2	75	=75/153	49.0%	X%
3	22	=22/153	14.4%	Y%

Total	**153**	**100%**	**100%**	**100%**

To calculate for X and Y we use the formula above:

- **X = (1-30%)/(1-36.6%)*49% = 0.7/0.634*.49 = 54.1%**
- **Y = (1-30%)/(1-36.6%)*14.4% = 0.7/0.634*.144 = 15.9%**

Save Personal Profile

The consumer will also be given the option to save their preference profile. They may want to do this for several reasons:

- To show to someone else and or/ merge with another person's profile
- To take a break and then search at a later time,
- To be able to repeat the search more than once (so they can investigate the returns at their leisure).

Note that this will require them to provide a user name/password and/or email address for verification.

Combining Results Across Multiple Users

The system also needs to be able to combine results across multiple users – from a segment as small as two users to the entire population of users. Combined results are used for three purposes:

1) Search for products based on a segment of users: What type of car would "Price Shoppers" want, or 'women under 25', or a husband and wife?
2) Perform standard types of conjoint analyses, such as relative importance, for the aggregate sample of users or for a particular segment.
3) Run product/market simulations using the aggregate or segment level data.

For the preferred method to combine two or more users is to merge their profiles into one composite profile. If another user has a saved profile, a consumer may want to merge the profiles to with his/her own profile to see how the different product attributes drive a unified buying strategy. For example, if a husband and a wife are looking for a car together, they may want to search individually, but they may also want to search jointly using a combined profile.

The math behind this is straightforward, simply taking the averages of the final optimal weight utilities for each level among all of the users whose profiles are to be

merged. Once new utility values representing the averages are calculated, scaled utilities and relative importances can be calculated as described earlier.

Search Product Database based on User Buying Profile and Report Results

The system must be capable of searching the product database and scoring products against the user's buying profile. Results will be provided in a rank-ordered list with a horizontal bar indicating the degree of fit. Product names and high-level descriptions are to be returned during this step. The user, or the system administrator, can define how many results are to be returned at once.

Product Scoring Algorithm

Search results will contain i) high-level descriptions and ii) 'best match scoring' based on the overall fit with the consumer's preference profile will be presented. Obviously, to retrieve and present this information, each product/service in the database will need to contain descriptors that can be scored against the criteria that define the individual's buying strategy.

To accomplish this, each feature of each product in the database is coded as a level corresponding to the conjoint. Once the conjoint is completed, the software simply adds the utility values for each level that makes up the product and scores that relative to the scores for each available product. For example, if the best product had an overall utility score of 74 and the second best a score of 71, then the first would receive a rating of 100% and the second a rating of 96%. Utility values would have to be estimated/interpolated for feature levels that were not expressly tested in the conjoint. The scoring algorithm would also be subject to some additional constraints:

1) It would need to account for constraints or unacceptables specified at the beginning of the interview.
2) An additional field may be added for manufacturer/merchant margin to help up-sell consumers. For example, if a consumer places the exact same utility on two products, we would want to recommend first the product that delivered the higher margin to the merchant. We may even have the software choose the ten most preferred and then select the five among those with the highest margins for presentation to the user.

Recall, from the example:

Attribute	Attribute Name	Level	Level Name	Final Scaled Utilities
1	*Brand*	1	Dell	56
		2	Compaq	51
		3	IBM	52
		4	Gateway	0
2	*Price*	5	$3,000	0
		6	$2,250	44
		7	$1,500	75
3	*Processor*	8	Pentium	0
		9	Pentium II	22

Now, suppose the database contained six product profiles:

First the utility of each product level would be calculated.

Product	Brand	Brand Utility	Price	Price Utility	Processor	Processor Utility
1	Dell	56	$3,000	0	Pentium	0
2	Compaq	51	$2,250	44	Pentium II	22
3	IBM	52	$1,500	75	Pentium	0
4	Gateway	0	$2,250	44	Pentium II	22
5	Compaq	51	$2,500	29*	Pentium	0
6	Toshiba	0**	$2,000	54**	Pentium	0

* Interpolated value – Since $2,500 is two-thirds of the way between $3,000 and $2,250 it receives two-thirds of the difference in value of 44 points. All values are rounded to the nearest whole integer.

** Since Toshiba was not one of the levels included in the conjoint it would receive a value equal to the lowest level (zero).

*** Interpolated value – Since $2,000 is one-third of the way between $2,250 and $1,500 it receives the value of 44 points (equal to $2,250) plus one-third of the difference in value between 44 and 75 points (31 point difference). All values are rounded to the nearest whole integer.

Second, the total utility would be calculated. This value would then be compared to the best product in the database to generate a recommendation score.

Product	Total Utility	Calculation	Recommendation Score
1	56	=56/127	44%
2	117	=117/127	92%
3	127	=127/127	100%
4	66	=66/127	52%
5	80	=80/127	63%
6	54	=54/127	43%

Provide Explanation of Product Scoring

The system must be capable of providing an explanation to the user of how the products were scored and how each product compares to the user's own buying profile. In one embodiment, this may be displayed using two dynamically generated charts side-by-side. The users preference profile will be displayed in one chart and the product's feature by feature fit with the user's preferences will be displayed in the other chart. The product fit will be calculated as follows:

Each attribute level that makes up the product will be assigned a utility score. If the attribute level is identical to one of the levels in the study it will be assigned that level's value, otherwise the value will be interpolated. The fit will then be the value of the particular product attribute level utility divided by the maximum attribute level utility in the product database for that specific product attribute. For example, if the product has a price of $1200 with a utility of 84, and the lowest price in the database is $1100 with a utility of 89, then the product's price fit score will be equal to 84/89, or a fit of 94.4%.

Regression Module

This module is called repeatedly to calculate regression coefficients that are also the utilities of the levels. The module needs to be sent two items: 1) a matrix of independent variables which indicate the presence or absence of each level in the product being evaluated and 2) a vector of dependent variables which represent the respondents' evaluation of the product with that combination of levels. The regression module returns the regression coefficients (utilities).

The general equation for multiple linear regression is:

(1) $Yi = b1\ X1i + \bullet\bullet\bullet\bullet\bullet + bk\ Xki$ for $i = 1$ to n

Where

Yi is the ith observation of the independent variable (to be explained)

Xki is the value of the kth dependent variable (the explanors)

b1 is the coefficient of the 1st of k dependent variables

Y and X are known. The b's are unknown. The statistical procedure solves the set of simultaneous equations for the values of b in terms of Y and X. Note: The unknown error term, usually written ui, has been omitted because we will not have to use the estimated values of the errors. These values are used to calculate various measures of how well the estimated values for b explain the variation in Y. In our application, we have to use the values we get for b instantly to choose the next question to ask the web visitor so we have no opportunity to get additional data or re-specify the equation to change the values for b and related goodness of fit statistics.

A constant term, usually written b0, is also omitted since it is not needed in the pairs section. The constant or intercept it set to zero when running the regression for the pairs. However, the constant term will be needed for the regression equations in the calibration section. The addition of the constant term does not change the formulas for the value of b (but it does change the values calculated). Note: As the regression used in the calibration section only has two independent variables, a formulaic approach will be used instead of the matrix algebraic approach used for the pairs regressions. Either approach will generate the same coefficient and intercept values for the calibration questions.

Our application is preferably programmed in matrix rather than scalar algebra (the formulaic approach) because of the potentially large number of independent variables (one for each attribute level). In other embodiments, scalar formulas as known in the art may be implemented.

Methodologies for regression are well known in the art as exhibited by Applied Linear Regression Models, Neter, Wasserman and Kutner, 1983, pp. 226-238, which is expressly incorporated herein by reference. The process of matrix multiplication is well know in the art as described in Algorithms, 2nd Ed., Robert Sedgewick, 1988, pp. 532 – 533, which is incorporated herein by reference. Similarly, the process of matrix inversion is also well known in the art, typically inverses may be calculated via Guassian elimination or Cramer's rule; these approaches are summarized in Computer Graphics: Principles and Practice, 2nd Ed., Foley, van Dam, Feiner and Hughes, 1990, pp. 1105-1106, which is incorporated herein by reference.

FIG. 1 displays the architecture of a typical system according to the present invention. A purchase decision support environment **100** will include a server **155a, 155b, 155c** and a data store **110** in communication with the server. The data store may include one or more database servers **160a, 160b, 160c** connect to a central or distributed storage system **165**. Alternative storage architectures for data store **110**, as will be known to those of skill in the art, are possible with in the scope of the present invention. Data store **110** communicates with server **155a, 155b, 155c** via ethernet **130** although in other embodiments the communication may be by other means such as Internet **120** or direct bus or serial connection. Distribution of sessions with purchasers **170a, 170b, 170c, 170d** from the user community **135** may occur across a set of servers **105**. In such an arrangement, the distribution in one embodiment could be managed by load balancing device **140**. The environment **100** may also include firewall **143** and router **146** for managing and controlling access to ethernet **130**; firewall **143** and router **146** may in some embodiments reside on the same hardware platform.

Purchasers **170a, 170b, 170c, 170d** communicate with a server (e.g. **155a**) through the Internet **120** and ethernet **130** although alternate communication channels may be used such as direct connection, dial-up to the server, or direction connection via ethernet **130** in other embodiments. Similarly, sellers and manufacturers **125** will

communicate with a server (e.g. **155a**) through the Internet **120** and ethernet **130**. Again, other communications channels may be used within the scope of the present invention.

The foregoing discussion makes reference to the PRECISION CHOICE software tool (Precision Choice hereinafter); PRECISION CHOICE is a trademark of Online Insight Incorporated (Atlanta, GA). Precision Choice is a preferred embodiment of the present invention. The details of this preferred embodiment are provided as exemplary. It will be readily appreciated that many deviations may be made from the specific embodiment disclosed in this specification without departing from the invention.

Precision Choice is the e-commerce equivalent of an expert salesperson in the traditional buying environment. Precision Choice interacts with a consumer to help him or her find the right solution for a purchasing decision. In a nutshell, Precision Choice educates the consumer, asks questions, analyzes responses, and presents ranked product recommendations.

Precision Choice also goes one important step further. The best salespeople learn everything they can about their customers and record this information so that they can serve them more effectively in the future. For each consumer, Precision Choice builds and stores a preference profile that analyzes how this person makes purchasing decisions.

During an interview, Precision Choice walks a consumer through a question and answer process. At the start of each interview, Precision Choice accepts a definition of the attributes that define the decision being made. This definition of attributes is called the study definition, or the study for short. For example, if the consumer is purchasing a laptop computer, attributes in the study definition might be hard drive size, screen size, and amount of RAM. Because Precision Choice can accept this information at the start of each interview, a single Precision Choice implementation can guide consumers through any number of different decisions.

For example, a Web site selling financial services could use a single Precision Choice implementation to help consumers choose mutual funds, credit cards, life insurance policies, and checking accounts. If the online merchant adds a

recommendation process for mortgages, absolutely no configuration changes or new software deployment within the Precision Choice engine are required to add a mortgage finder. A travel site could use a single implementation of Precision Choice to guide consumers to the right cruise, airline, honeymoon package, and/or hotel.

The number of questions and types of attributes can be different for every consumer, so novices can be treated differently than experienced buyers. A consumer can even specify which attributes matter and which attributes do not matter, effectively controlling the kinds of questions Precision Choice asks in the interview.

Overview of the Interview Process

The FIG. 6 illustrates the process of using Precision Choice to conduct an interview. Note that the Figure shows the interaction of two software systems: Precision Choice and an external system that feeds information to Precision Choice and receives product recommendations. Note that certain information is fed into Precision Choice and that Precision Choice outputs certain information. To provide maximum flexibility, all information input into or output from Precision Choice is communicated in an XML format, although other formats may be used in alternate embodiments. This allows ease of integration between Precision Choice and most any other software system.

The recommendation engine accepts inputs and outputs in terms of an XML language called the Precision Choice XML Command Language. This XML language creates a standard, documented API by which other software systems can interact with Precision Choice as more fully described below, yielding a tremendous amount of flexibility. The Precision Choice presentation layer does the job of transforming the XML output from the Precision Choice recommendation engine into an attractive set of screens, which serve as the user-interface. The presentation layer also receives input from the user, translates it into XML, and sends it as input to the Precision Choice engine. Because these two layers are cleanly separated, it is possible to use an external or custom user-interface layer if desired.

For example, a bank might have information that indicates which type of checking account a consumer prefers, and, therefore, might know the answer to some questions that Precision Choice generates without having to ask the individual

consumer. Since Precision Choice outputs questions via XML, the bank can have a software system intercept some questions, answer the questions, and supply the responses to Precision Choice (again via XML). By implementing such a process, the bank avoids asking questions for which it already knows the answers, saving time for the consumer.

Note also that at the start of each interview a product catalog is provided to Precision Choice. As with all other forms of input into Precision Choice, this catalog is specified in terms of XML. Specifying this information at the start of each interview results in a tremendous amount of flexibility. Some products may not be available to certain consumers, or certain special products may be available only to select individuals.

In some cases, product pricing and other product attributes may vary from individual to individual. In addition, some products change almost continually: i.e. mortgage rates are updated as often as every 15 minutes. Precision Choice can accommodate all of these situations because the product catalog's definition catalog can be different for each interview. For product catalogs reused in different interviews, it is possible to give Precision Choice the catalog definition once and then to refer to it for subsequent interviews.

Once the study definition and the product catalog definition are provided, the interview can occur. Precision Choice starts the interview process by generating the questions it needs to ask the consumer. Precision Choice delivers these questions in an XML format so that another software system can parse these questions, present them to the consumer, and submit the responses back to Precision Choice via XML. A presentation layer that utilizes Java Servlets and Java Server Pages generates HTML representations of these questions and handles the interaction with the consumer. After the consumer responds to all questions, Precision Choice provides a preference profile for this consumer and provides product recommendations. Precision Choice records the preference profile in the preferences database, and the interview is then complete.

Technical Overview of Precision Choice

Precision Choice is designed to meet the demands of e-commerce:

It offers platform independence because it was developed in Java 2

It has powerful built-in load balancing features that result in high availability and numerous options for improving performance and handling load

It offers ease of integration with other software systems because of its commitment to open standards such as XML and CORBA

As noted above, all input into and output from Precision Choice is communicated in terms of an XML language called Precision Choice Command XML.

Precision Choice offers a flexible load-balancing scheme. It is easy to configure clusters of Precision Choice servers to work together to handle the load required for a deployment. Each cluster can operate on a single piece of hardware or be spread across multiple pieces of hardware.

Precision Choice is capable of interacting with external systems via CORBA, Enterprise JavaBeans, Java RMI, Microsoft COM/DCOM, HTTP, or virtually any other communication protocol. Precision Choice defines the specifications for an adaptor through which it communicates with external systems. When Precision Choice communicates with a Java/ RMI-based software system, an adaptor may not be necessary. For communicating with external systems based on other technologies, it is a simple process to develop a custom adaptor.

Precision Choice is implemented entirely within Java 2. While the system should be compatible with any platform that can host a Java 2 virtual machine, it has been tested and certified on Solaris 2.6 and Windows NT.

Precision Choice uses JDBC to communicate with the preferences database and to store study definitions and product catalog definitions. Precision Choice has been tested and certified to work with Oracle 8.1. Any database that can communicate via JDBC or ODBC should be compatible with Precision Choice, and, therefore, other databases may be supported after sufficient testing.

As mentioned earlier, the preferences data that Precision Choice collects offers a detailed understanding of the consumer's preferences and can be used for purposes such as the following:

Predicting the market share of potential product introductions

Measuring brand equity

Many other analyses

Interview Process

To offer successful client solutions based on Precision Choice, it is necessary to understand the rationale behind the questions that Precision Choice asks consumers. The foregoing discussion asks and explains what Precision Choice learns from the consumers' responses through the use of two examples: helping a consumer identify the best mutual fund and helping a consumer identify the right laptop computer.

Precision Choice Question Types

Ratings Questions

To identify the best mutual fund, Precision Choice needs to know how a consumer feels about the various mutual fund categories. One person may prefer Aggressive Growth funds and loathe Bond funds: another person may have completely divergent feelings. To uncover this information, Precision Choice asks a rating question for Category. This question might look something like the screen capture seen in FIG. 7:

Precision Choice uses a rating question when it needs to discover how a consumer feels about each level of one specific product feature. In this example, Precision Choice learns that "Aggressive Growth" and "Growth" are tied for this consumer's favorite Category, that "Small Company" is third, and that the consumer does not find the other categories of mutual funds appealing.

Comsumers should answer ratings questions from the perspective of "all else being equal." In other words, a consumer ideally looks at the question in the screenshot above and thinks, "All else being equal, I prefer Aggressive Growth and Growth mutual funds."

Precision Choice does not want a consumer to think something like the following: "Since International stock funds tend to have a higher Load, I'll rate International Stock as an unattractive Category." The Precision Choice engine would interpret this response as if the consumer said, "If I were presented with two mutual funds that were exactly the same in every way, but one was an International Stock fund and the other was an Aggressive Growth fund, I would prefer the Aggressive Growth fund four times more than I would the International Stock fund." Precision Choice uses

this information as part of the input it needs to calculate a numerical score representing how appealing each Category is to this consumer.

Ratings questions are only necessary for attributes that different consumers value differently, i.e., for attributes that are subjective. Brand is a good example, as people have different feelings about brands.

For some attributes, Precision Choice does not ask a rating question. For example, Precision Choice does not ask a rating question about price because, everything else being equal, virtually everyone prefers a lower price. If Precision Choice presents a consumer with two laptops that are alike in every way (same brand, same hard drive capacity, etc.) except that one costs $3500 and the other costs $3000, almost any consumer would prefer the one that costs $3000. In the mutual fund arena, Load is a good example. If minimum investment, Morningstar ratings, and returns are the same for two mutual funds, almost anyone would prefer the mutual fund with a lower Load. Precision Choice skips the rating question for these types of attributes, saving time for the consumer.

Attributes that do not vary in appeal from consumer to consumer - and therefore do not require a rating question - are called a priori attributes. Often, price is an a priori attribute. For laptops, hard drive capacity and RAM are a priori attributes. For mutual funds, Three-Year Return and Load are a priori attributes.

For the rating questions, Precision Choice accepts responses as numeric values between 1 and 5, inclusive. Ties are allowed.

Importance Questions

Precision Choice next needs to learn what importance a consumer places on certain attributes. For example, some people could care less about Morningstar ratings, but are very sensitive to Load. Others are not Load-sensitive, but place special emphasis on Three-Year Return. To generate relevant product recommendations, Precision Choice needs to learn what importance a consumer places on each attribute.

The basic kinds of questions Precision Choice asks are like the following:

- How important to you is getting the Category that you want?
- How important is Load to you?

Precision Choice goes one step further to make such questions more precise, as the following example illustrates.

Imagine that a consumer has a budget of $5000 to buy a laptop. When Precision Choice asks, "How important to you is price?", the consumer might respond that price is extremely important. There is $5,000 to spend, and anything over $5,000 is over budget. However, if the consumer knows that prices of the laptops in the product database range from $2,000 to $4,500, he or she would probably respond that price is somewhat important, but not extremely important, since all the laptops are within budget.

As this example illustrates, a precise importance question is framed in terms of the best and worst values. Thus, Precision Choice includes the best and worst setting for each attribute when presenting an importance question. For price, Precision Choice would ask, "How important to you is price, considering that the laptops range in price from $2,000 - $4,500?" For RAM, Precision Choice would ask, "How important to you is RAM, considering that the laptops range from 16 MB to 128 MB?"

There is an interesting ramification of supplying the best and worst values with importance questions: Precision Choice cannot generate importance questions until it receives responses to its ratings questions. Based on the consumer's previous responses, Precision Choice would ask an importance question for Category that might look like the following, "How important to you is Category, considering that the least desirable Category available is International Stock, and the most desirable Category is Aggressive Growth?" In other words, importance questions for non-a priori attributes are also expressed in terms of the best and worst values. Precision Choice does not know the best and worst values for non-a priori attributes until it receives a consumer's response to a rating question.

FIG. 8 illustrates how Precision Choice might present an importance question to a consumer:

From the consumer's response to this question, Precision Choice learns what weight to place on each attribute, relative to the other attributes. Precision Choice uses this information to calculate a numerical score that indicates how much appeal each possible setting for each attribute has for this consumer.

For the importance questions, Precision Choice accepts responses as numeric values between 1 and 4, inclusive: 1 is the worst rating, and 4 is the best rating.

Pairs Questions

The next kind of question that Precision Choice generates is called a pairs question. This question type affords Precision Choice much of its precision.

When people express what they want, a natural tendency is to rate everything as being important. After all, consumers really want a laptop computer that is their favorite brand, with the greatest amount of RAM and hard drive space available on the market, for less than $50! Many consumers would express their overt preferences in such terms. In our universe of limited resources, however, such products do not exist.

Pairs questions are so important because they force the consumer to evaluate trade-offs. Pairs questions often take the following form: "If you were forced to choose between a bigger hard drive with a higher price or a smaller hard drive with a lower price, which would you choose?"

To build pairs questions, Precision Choice generates hypothetical products and displays them side-by-side. Each hypothetical product is built so that it has some desirable features and some undesirable features. Precision Choice builds the pairs questions so that the hypothetical product on the left is similar in appeal to the hypothetical product on the right.

For example, consider the pairs question in the screenshot seen in FIG. 9. The hypothetical product on the left has a lower Load, which is desirable, and a lower Three-Year Return, which is undesirable. All pairs questions present the consumer with such a trade-off decision. Precision Choice is designed to generate pairs questions that are as difficult as possible because the answers to difficult questions teach Precision Choice the most about the consumer's preferences. For each pairs question, Precision Choice builds two hypothetical products as similar as possible in appeal to the consumer, displays the two products side-by-side, and asks the consumer to choose.

For the pairs questions, Precision Choice asks the consumer to respond along a nine-point scale, ranging from "Strongly prefer left product" to "Neutral" to "Strongly prefer right product." This nine-point scale enables Precision Choice to quantify the consumer's preferences precisely.

In the example above, if the consumer clicks on "Strongly prefer right product," Precision Choice learns that Three-Year Return is much more important than Load. On the other hand, if the consumer clicks on the sixth radio button (just to the right of neutral), this reflects the consumer's slight preference for the product on the right. In this case, Precision Choice learns that Three-Year Return is slightly more important than Load. Either way, Precision Choice precisely quantifies how strongly or slightly the consumer prefers a product feature.

In the example above, the pairs question has only two attributes: Three-Year Return and Load. The trade-off in the pairs question directly pits these two attributes against each other. However, pairs questions are not limited to two attributes.

By default, Precision Choice asks seven pairs questions. The first few pairs questions are limited to two attributes. The next few pairs questions have three attributes. The final pairs question has four attributes. The questions that Precision Choice asks the consumer become more complex as the number of attributes increases.

The sample question in the screenshot of FIG. 10 is a pairs question with four attributes.

Pairs questions are generated dynamically. Precision Choice only generates one pairs question at a time. The system receives a response to a pairs question, interprets the response, refines its understanding of the consumer's preferences based upon the response, and uses its updated understanding to decide how to build the next pairs question. Each consumer sees pairs questions that are relevant directly to him or her, and the pairs questions grow increasingly more difficult as the system hones in on the consumer's preferences. To generate pairs questions, Precision Choice freely combines various product features to produce tough trade-offs. In this way, each consumer's experience with Precision Choice is unique.

Although the values that Precision Choice uses to create hypothetical products are representative of the acceptable range of values for each attribute, it is important to emphasize that the pairs questions present hypothetical products. The example in Figure 3 (above) shows a fund on the left side with a Load of 6% and a Three-Year Return of 30%; there may not actually be a product in the database with these features.

Likewise, there may not be a mutual fund in the product database that matches the features displayed on right side: 0% Load for a Three-Rear Return of 0%.

Whether or not matching products exist is irrelevant. Precision Choice is trying to learn how the consumer trades off various product features against each other.

Calibration Questions

By this point in an interview, Precision Choice has an excellent understanding of the consumer's preferences. As noted above, one of the benefits of Precision Choice is that it stores preference profiles in a preferences database. This preference profile can be used at both individual and aggregate levels to guide important marketing decisions.

It is possible that a consumer might stop paying attention to the questions during an interview. The interview's results would be inaccurate because of the inconsistency in the answers. For subsequent analysis of the preference data Precision Choice collects, it is helpful to be able to identify interviews with inconsistencies and to discard their results, or at least to weigh them appropriately.

To address this concern, Precision Choice offers an optional question type called calibration questions. Precision Choice asks calibration questions to determine how consistent a consumer is throughout the course of an interview. Responses to calibration questions can also indicate which consumers are "just looking" and which are definitely "in the market." A sample calibration question appears in the screen shot of FIG. 11.

Precision Choice does not ask calibration questions by default. Most business situations require the shortest possible process for the end-user. Since calibration questions do not increase the actual precision of the recommendations, they can be eliminated from the Precision Choice process. However, some situations require greater precision in the preference data that Precision Choice gathers, even at the expense of an extra step for the consumer.

When calibration questions are active, Precision Choice asks four calibration questions in it spreferred embodiment, but Precision Choice can be configured to ask fewer calibration questions. Each question presents the consumer with a hypothetical product and asks how likely it is that he or she would purchase that product on a scale from 0% to 100%. One calibration question presents a hypothetical product that,

according to the preference profile, Precision Choice knows is unappealing to the consumer. Another hypothetical product is one that Precision Choice knows is appealing to the consumer. The other two calibration products are somewhere between these two.

Precision Choice's detection of inconsistency is quite precise. If the consumer indicates that he or she is more likely to purchase the unappealing product than the appealing product, Precision Choice detects inconsistency and quantifies how different the consumer's responses to the calibration questions are from expected responses: the resulting numerical value indicates the level of consistency in the responses.

A Sample Precision Choice Interview

This sample interview shows how the concepts discussed above work in practice and illustrates how Precision Choice interacts with a consumer. This interview's purpose is to identify a mutual fund that meets the consumer's preferences. In this example, we consider the following attributes of mutual funds.

Category. Mutual funds are grouped into categories that express each fund's overall investment philosophy at the most fundamental level: Aggressive Growth, Growth, Small Company, International Stock, Balanced, Equity Income, Corporate Bond, and Municipal Bond.

Morningstar Rating. Morningstar is an independent rating service that rates mutual funds on a scale of 1 to 5, based on risk-adjusted return. Funds are rated in comparison with other similar funds broken into US Stock, International Stock, Taxable Bond, and Tax-Free Bond. The top 10% of funds receive a rating of five, the next 22.5% receive a four, the middle 35% receive a three, the next 22.5% a two, and the bottom 10% receive one. Values for this attribute range are whole numbers from 1 through 5.

Morningstar Risk. Morningstar is an independent rating service that gives every fund a risk rating. In this case, risk is a measurement of fund under-performance in the past, i.e., returns less than those that could be achieved by investing in a Treasury bill. A fund that has under-performed during many months of recent years receives a high risk rating. Values for this attribute range from .5 through 1.5.

Three-Year Return. This is the average annual amount of value that the funds have created, either through income or capital growth, over the three years from March 1996 to March 1999, measured as a percentage of initial investment. These values have an infinite possibility range, but typical numbers range from 0% - 30%.

Load. This is a percentage of the initial investment charged up-front before any money is invested. For example, if $10,000 is invested in a fund with a 2% Load, then $200 is charged up front, leaving $9,800 to be invested. Load charges typically range from 0% to 6%.

The above information is the kind of information that makes up the study definition, as explained above. Given this study definition, Precision Choice would ask the consumer the following questions in the interview.

First, Precision Choice needs to know how this consumer feels about the various mutual fund categories. Category is a non-a priori attribute, and, therefore, Precision Choice asks a ratings question to learn about the consumer's preferences related to Category. This question might look like the sample in the screenshot of FIG. 12.

In this study, there are no remaining non-a priori attributes. The remaining attributes (Morningstar Rating, Morningstar Risk, Three-Year Return, and Load) are all a priori. Precision Choice does not ask ratings questions for the a priori attributes.

Precision Choice then asks importance questions to learn what importance the consumer places on each attribute. In this study, the next five questions would be importance questions. Precision Choice generates these five questions at one time. The sample screen shot of FIG. 13shows all five questions presented on one screen. Note that it displays the best and worst values for each product.

Precision Choice then moves into the pairs questions. Precision Choice analyzes what it knows of the consumer's preferences to this point and uses that information to build the toughest trade-off question it can. The number of attributes in each pair is configurable, but in the preferred embodiment, the first pairs question contains only two attributes. FIG. 14 illustrates how Precision Choice might present this first pairs question.

After Precision Choice receives the consumer's response to the first pairs question, it adjusts the preference profile to incorporate what it has learned. Precision Choice again builds the toughest trade-off question that it can. The second pairs question for the consumer also contains two attributes. FIG. 15 illustrates how Precision Choice might present this question to a consumer.

Precision Choice generates the third pairs question after it receives the response to the second pairs question and incorporates it into the preference profile. At this point, Precision Choice presents pairs questions with three attributes each, as in the sample screenshot of FIG. 16.

The fourth pairs question also has three attributes. The trade-offs are becoming harder because Precision Choice is honing in precisely on the consumer's preferences, as seen in FIG. 17. The fifth pairs question has three attributes, as illustrated in FIG. 18. Further, the sixth pairs question also has three attributes, as depicted in the screen capture of FIG. 19. Precision Choice then generates the seventh pairs question, the only pairs question with four attributes. FIG. 20 displays a sample seventh pairs question.

Precision Choice would then ask calibration questions, if configured to do so. Calibration questions are not asked by default and, therefore, are not included in this sample interview.

At this point, Precision Choice's interview process is complete, and Precision Choice is ready to provide product recommendations.

Introduction to Precision Choice XML Command Language

This foregoing provides an introduction to the Precision Choice XML command language by examining the sample interview discussed above.

Overview of XML

All interaction with Precision Choice is accomplished through use of the Precision Choice command language. Commands are expressed using XML, a technology that allows data portability across platforms and applications. With XML, a source can provide data and also indicate what the data means in the document. Any process that can parse XML can understand the document.

With XML, you define your own tags and then provide the values for those tags. An XML document can define whatever tag is needed to describe the data accurately. XML is a textual document made up of elements and data. Each element can contain other elements, data, or nothing. An element can also contain parameters, called attributes, that further define an element. Since the example contains a tag called "attribute," the text refers to an element's attributes as parameters. An XML snippet that defines a product in Precision Choice follows:

```
<product id="1" name="Fund A">
<attribute id="Category">Aggressive Growth</attribute>
<attribute id="Morningstar Rating">5</attribute>
<attribute id="Morningstar Risk">1.2</attribute>
<attribute id="One Year Return">66.5</attribute>
<attribute id="Three Year Return">39.0</attribute>
<attribute id="Load">0.0</attribute>
<attribute id="Minimum Investment">2500</attribute>
</product>
```

In the code above, the product is the first element defined:

```
<product id="1" name="Fund A">
```

The element contains multiple attributes. This one has two parameters: an id (id="1") and a name (name="Fund A"). Each attribute contains text for the attribute and has one parameter: id (id="Category"). The first attribute defined for this product is Category: "Aggressive Growth."

A document tag definition file (dtd) ensures that an XML document provides all required information required. An example dtd for the above product would be the following:

```
<!ELEMENT product ( attribute+ ) >
<!ATTLIST product id  NMTOKEN  #REQUIRED>
<!ATTLIST product name CDATA   #REQUIRED>

<!ELEMENT attribute  ( #PCDATA )>
<!ATTLIST attribute id  CDATA   #IMPLIED>
```

This dtd specifies that there is an element called product and that it contains an element called attribute:

<!ELEMENT product (attribute+) >

The plus (+) sign after attribute indicates that there must be at least one attribute for the product. The product has two parameters called id and name:

<!ATTLIST product id CDATA #REQUIRED>

<!ATTLIST product name CDATA #REQUIRED>

The name and id parameters are defined as CDATA, which means that this parameter can contain any type of text. Both parameters are required.

The next element defined is the attribute element. This element can contain PCDATA, which means the element can contain any type of text. It has one attribute called id that is not required (#IMPLIED) and is of type CDATA. An application that processes XML data uses a dtd file to ensure that the XML data are correct.

XML documents are case-sensitive. The tags <PRODUCT> and <product> are different. If the <PRODUCT> tag were in an XML document and the above dtd was used to validate it, there would be an error: the dtd defines <product>, not <PRODUCT>.

There are many books and Web resources devoted to XML. This overview only begins to define what you can accomplish with XML. Websites such as www.xml.com and msdn.microsoft.com/xml contain extensive information on XML.

Overview of the Sample Interview

The sample interview above addresses finding a mutual fund that is the best match for an end-user. The interview is based upon the following study definition:

Category: Mutual funds are grouped into categories that express each fund's overall investment philosophy at the most fundamental level. Values for this attribute include Aggressive Growth, Growth, Small Company, International Stock, Balanced, Equity Income, Corporate Bond, and Municipal Bond.

Morningstar Rating: Morningstar is an independent rating service that rates mutual funds on a scale of 1 to 5, based on risk-adjusted return. Funds are rated in comparison with similar funds: US Stock, International Stock, Taxable Bond, and Tax-Free Bond. The top 10% of funds receive a rating of 5, the next 22.5% receive a 4, the

middle 35% receive a 3, the next 22.5% a 2, and the bottom 10% receive 1. Values for this attribute range are whole numbers from 1 through 5.

Morningstar Risk: Morningstar is an independent rating company that gives every fund a risk rating. Risk is a measure of fund under-performance in the past, i.e., returns less than those that could be achieved by investing in a Treasury bill. A fund that has under-performed during many months of recent years will receive a high risk rating. Values for this attribute range from .5 through 1.5.

Three-Year Return: This is the average annual amount of value that the fund has created, either through income or capital growth, over the three years from March 1996 to March 1999, measured as a percentage of initial investment. These values have an infinite possibility range, but typical numbers range from 0% - 30%.

Load: This is a percentage of the initial investment charged before any money is invested. For example, if $10,000 is invested in a fund with a 2% load, then $200 is charged up front, leaving $9,800 to be invested. Load charges typically range from 0% to 6%.

Defining the Study and the Product Catalog

The XML below represents this study:

```
<study name="mutual funds" version="1.0">
<attribute desc="Category" unit="text">
<level>Aggressive Growth</level>
<level>Growth</level>
<level>Small Company</level>
<level>International Stock</level>
<level>Balanced</level>
<level>Equity Income</level>
<level>Corporate Bond</level>
<level>Municipal Bond</level>
</attribute>
<attribute desc="Morningstar Rating" unit="numeric"
interpolate="no" apriori="yes">
<level>5</level>
```

<level>4</level>
<level>3</level>
<level>2</level>
<level>1</level>
</attribute>
<attribute desc="Morningstar Risk"
unit="numeric"
interpolate="yes"
apriori="yes">
<level>0.5</level>
<level>0.75</level>
<level>1.0</level>
<level>1.25</level>
<level>1.5</level>
</attribute>
<attribute desc="Three Year Return"
unit="percent"
interpolate="yes"
apriori="yes">
<level>30</level>
<level>20</level>
<level>10</level>
<level>0</level>
</attribute>
<attribute desc="Load"
unit="percent"
interpolate="yes"
apriori="yes">
<level>0</level>
<level>2</level>
<level>4</level>

<level>6</level>

</attribute>

</study>

To define a study, you specify the attributes and levels of a particular type of product, in this case of mutual funds. There are three elements in the study definition. Study, the first element, must contain at least three attribute elements and cannot contain more than nine elements. The study has two parameters: name and version. Both elements are used to identify the study. They are required, but are not used in Precision Choice processing.

There are four parameters that Precision Choice uses to determine when and how to present an attribute in the questions. The parameter desc contains text that should be displayed to the user when this attribute is present in a question. This alphanumeric parameter is required. The unit parameter determines how this attribute is formatted. The value for this parameter must be defined in the units.xml file. For details, see Appendix A.

Interpolate and apriori, the last two parameters, are not required. The default value for each parameter is "No." Interpolate tells Precision Choice whether the attribute levels can be interpolated. For example, the attribute Morningstar Rating cannot be interpolated. This attribute has five distinct levels. The attribute Category cannot be interpolated and is not a priori. A priori tells Precision Choice that there is a preferred order for the attribute's values.

Each level defines a different value for an attribute. The order in which the levels are defined determines each level's rank for an a priori attribute. The first level defined is the best level, and the last level defined is the worst level. Values for the level should be defined so that they cover all values for an attribute. In the case of Load, values could range between 0 and 6. Hence, that level has four equidistant values: the best level is 0, and the worst is 6.

The goal of this interview is to provide mutual fund product recommendations for an end-user. To accomplish this, products are defined for Precision Choice using the following command.

<catalog name="mutualfunds" version="1.0">

<product id="1" name="Fund A">
<attribute id="Category">Aggressive Growth</attribute>
<attribute id="Morningstar Rating">5</attribute>
<attribute id="Morningstar Risk">1.2</attribute>
<attribute id="One Year Return">66.5</attribute>
<attribute id="Three Year Return">39.0</attribute>
<attribute id="Load">0.0</attribute>
<attribute id="Minimum Investment">2500</attribute>
</product>
<product id="4" name="Fund B">
<attribute id="Category">Aggressive Growth</attribute>
<attribute id="Morningstar Rating">5</attribute>
<attribute id="Morningstar Risk">1.4</attribute>
<attribute id="One Year Return">47.4</attribute>
<attribute id="Three Year Return">28.1</attribute>
<attribute id="Load">5.0</attribute>
<attribute id="Minimum Investment">0</attribute>
</product>
<product id="11" name="Fund C">
<attribute id="Category">Growth</attribute>
<attribute id="Morningstar Rating">5</attribute>
<attribute id="Morningstar Risk">0.9</attribute>
<attribute id="One Year Return">79.9</attribute>
<attribute id="Three Year Return">47.9</attribute>
<attribute id="Load">0.0</attribute>
<attribute id="Minimum Investment">2500</attribute>
</product>
<product id="14" name="Fund D">
<attribute id="Category">Growth</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">1.2</attribute>

<attribute id="One Year Return">63.3</attribute>
<attribute id="Three Year Return">31.8</attribute>
<attribute id="Load">5.5</attribute>
<attribute id="Minimum Investment">1000</attribute>
</product>
<product id="31" name="Fund E">
<attribute id="Category">Small Company</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">1.9</attribute>
<attribute id="One Year Return">80.9</attribute>
<attribute id="Three Year Return">32.7</attribute>
<attribute id="Load">5.0</attribute>
<attribute id="Minimum Investment">1000</attribute>
</product>
<product id="32" name="Fund F">
<attribute id="Category">Small Company</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">1.7</attribute>
<attribute id="One Year Return">38.5</attribute>
<attribute id="Three Year Return">31.6</attribute>
<attribute id="Load">0.0</attribute>
<attribute id="Minimum Investment">5000</attribute>
</product>
<product id="41" name="Fund G">
<attribute id="Category">Equity Income</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">0.8</attribute>
<attribute id="One Year Return">28.3</attribute>
<attribute id="Three Year Return">22.3</attribute>
<attribute id="Load">0.0</attribute>
<attribute id="Minimum Investment">1000</attribute>

```
</product>
<product id="43" name="Fund H">
<attribute id="Category">Equity Income</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">0.8</attribute>
<attribute id="One Year Return">15.2</attribute>
<attribute id="Three Year Return">26.5</attribute>
<attribute id="Load">5.8</attribute>
<attribute id="Minimum Investment">500</attribute>
</product>
<product id="51" name="Fund I">
<attribute id="Category">International Stock</attribute>
<attribute id="Morningstar Rating">2</attribute>
<attribute id="Morningstar Risk">1.6</attribute>
<attribute id="One Year Return">83.4</attribute>
<attribute id="Three Year Return">-0.8</attribute>
<attribute id="Load">3.0</attribute>
<attribute id="Minimum Investment">2500</attribute>
</product>
<product id="52" name="Fund J">
<attribute id="Category">International Stock</attribute>
<attribute id="Morningstar Rating">1</attribute>
<attribute id="Morningstar Risk">3.2</attribute>
<attribute id="One Year Return">41.6</attribute>
<attribute id="Three Year Return">-19.2</attribute>
<attribute id="Load">0.0</attribute>
<attribute id="Minimum Investment">1000</attribute>
</product>
<product id="81" name="Fund K">
<attribute id="Category">Balanced</attribute>
<attribute id="Morningstar Rating">3</attribute>
```

<attribute id="Morningstar Risk">0.7</attribute>
<attribute id="One Year Return">30.7</attribute>
<attribute id="Three Year Return">21.2</attribute>
<attribute id="Load">5.0</attribute>
<attribute id="Minimum Investment">0</attribute>
</product>
<product id="82" name="Fund L">
<attribute id="Category">Balanced</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">0.5</attribute>
<attribute id="One Year Return">29.5</attribute>
<attribute id="Three Year Return">24.6</attribute>
<attribute id="Load">5.5</attribute>
<attribute id="Minimum Investment">500</attribute>
</product>
<product id="83" name="Fund M">
<attribute id="Category">Balanced</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">0.5</attribute>
<attribute id="One Year Return">28.2</attribute>
<attribute id="Three Year Return">24.3</attribute>
<attribute id="Load">0.0</attribute>
<attribute id="Minimum Investment">2500</attribute>
</product>
<product id="91" name="Fund N">
<attribute id="Category">Corporate Bond</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">0.9</attribute>
<attribute id="One Year Return">11.4</attribute>
<attribute id="Three Year Return">10.0</attribute>
<attribute id="Load">3.8</attribute>

<attribute id="Minimum Investment">2000</attribute>
</product>
<product id="92" name="Fund O">
<attribute id="Category">Corporate Bond</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">1</attribute>
<attribute id="One Year Return">8.2</attribute>
<attribute id="Three Year Return">8.7</attribute>
<attribute id="Load">0.0</attribute>
<attribute id="Minimum Investment">2000</attribute>
</product>
<product id="111"
name="CitiFunds National Tax-Free Income A">
<attribute id="Category">Municipal Bond</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">1.1</attribute>
<attribute id="One Year Return">8.3</attribute>
<attribute id="Three Year Return">0.3</attribute>
<attribute id="Load">4.5</attribute>
<attribute id="Minimum Investment">1000</attribute>
</product>
<product id="112"
name="Delaware National High Yield Municpals A">
<attribute id="Category">Municipal Bond</attribute>
<attribute id="Morningstar Rating">5</attribute>
<attribute id="Morningstar Risk">0.2</attribute>
<attribute id="One Year Return">6.6</attribute>
<attribute id="Three Year Return">8.2</attribute>
<attribute id="Load">3.8</attribute>
<attribute id="Minimum Investment">1000</attribute>
</product>

<product id="118" name="Strong Municipal Bond">
<attribute id="Category">Municipal Bond</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">0.7</attribute>
<attribute id="One Year Return">6.2</attribute>
<attribute id="Three Year Return">8.4</attribute>
<attribute id="Load">0.0</attribute>
<attribute id="Minimum Investment">2500</attribute>
</product>
</catalog>

The catalog element defines a group of products for Precision Choice. It must contain at least one product. There is no upper limit to the number of products that can be defined in a catalog. The catalog has two parameters: name and version. Both elements are used to identify the catalog. They are required, but are not used in Precision Choice processing.

A product is defined by multiple attributes, and a product must contain a value for each attribute defined in the study. The product definition may define attributes not in the study. These attributes are not part of the interview process, but they are returned in the Product Recommendations. The product element contains two parameters: id and name. The id parameter is the product key. It must be unique in the catalog and is required. The name parameter is the display value of the product. It is required.

Each product attribute is defined by one parameter and contains the value of the product attribute. The id parameter must match the desc parameter of the attribute in the study command.

The study and catalog must be loaded into Precision Choice before the interview begins. To accomplish this, the following XML command must be sent into Precision Choice.

<BATCHSET>
<SETSTUDY CID="1" NAME="mutualfunds">
<study name="mutual funds" version="1.0">
<attribute desc="Category" unit="text">

<level>Aggressive Growth</level>
<level>Growth</level>
<level>Small Company</level>
<level>International Stock</level>
<level>Balanced</level>
<level>Equity Income</level>
<level>Corporate Bond</level>
<level>Municipal Bond</level>
</attribute>
<attribute desc="Morningstar Rating"
unit="numeric"
interpolate="no"
apriori="yes">
<level>5</level>
<level>4</level>
<level>3</level>
<level>2</level>
<level>1</level>
</attribute>
<attribute desc="Morningstar Risk"
unit="numeric"
interpolate="yes"
apriori="yes">
<level>0.5</level>
<level>0.75</level>
<level>1.0</level>
<level>1.25</level>
<level>1.5</level>
</attribute>
<attribute desc="Three Year Return"
unit="percent"

```
interpolate="yes"
apriori="yes">
<level>30</level>
<level>20</level>
<level>10</level>
<level>0</level>
</attribute>
<attribute desc="Load"
unit="percent"
interpolate="yes"
apriori="yes">
<level>0</level>
<level>2</level>
<level>4</level>
<level>6</level>
</attribute>
</study>
</SETSTUDY>
<SETCATALOG CID="2" NAME="mutualfunds">
<catalog name="mutualfunds" version="1.0">
<product id="1" name="Janus Olympus">
<attribute id="Category">Aggressive Growth</attribute>
<attribute id="Morningstar Rating">5</attribute>
<attribute id="Morningstar Risk">1.2</attribute>
<attribute id="One Year Return">66.5</attribute>
<attribute id="Three Year Return">39.0</attribute>
<attribute id="Load">0.0</attribute>
<attribute id="Minimum Investment">2500</attribute>
</product>
<product id="4" name="Alger Capital Appreciation B">
<attribute id="Category">Aggressive Growth</attribute>
```

<attribute id="Morningstar Rating">5</attribute>
<attribute id="Morningstar Risk">1.4</attribute>
<attribute id="One Year Return">47.4</attribute>
<attribute id="Three Year Return">28.1</attribute>
<attribute id="Load">5.0</attribute>
<attribute id="Minimum Investment">0</attribute>
</product>
<product id="11" name="Janus Twenty">
<attribute id="Category">Growth</attribute>
<attribute id="Morningstar Rating">5</attribute>
<attribute id="Morningstar Risk">0.9</attribute>
<attribute id="One Year Return">79.9</attribute>
<attribute id="Three Year Return">47.9</attribute>
<attribute id="Load">0.0</attribute>
<attribute id="Minimum Investment">2500</attribute>
</product>
<product id="14" name="WM Growth A">
<attribute id="Category">Growth</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">1.2</attribute>
<attribute id="One Year Return">63.3</attribute>
<attribute id="Three Year Return">31.8</attribute>
<attribute id="Load">5.5</attribute>
<attribute id="Minimum Investment">1000</attribute>
</product>
<product id="31" name="TCW/DW Mid-Cap Equity B">
<attribute id="Category">Small Company</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">1.9</attribute>
<attribute id="One Year Return">80.9</attribute>
<attribute id="Three Year Return">32.7</attribute>

<attribute id="Load">5.0</attribute>
<attribute id="Minimum Investment">1000</attribute>
</product>
<product id="32"
name="Robertson Stephens Emerging Growth A">
<attribute id="Category">Small Company</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">1.7</attribute>
<attribute id="One Year Return">38.5</attribute>
<attribute id="Three Year Return">31.6</attribute>
<attribute id="Load">0.0</attribute>
<attribute id="Minimum Investment">5000</attribute>
</product>
<product id="41" name="Value Line Income">
<attribute id="Category">Equity Income</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">0.8</attribute>
<attribute id="One Year Return">28.3</attribute>
<attribute id="Three Year Return">22.3</attribute>
<attribute id="Load">0.0</attribute>
<attribute id="Minimum Investment">1000</attribute>
</product>
<product id="43" name="GE Value Equity A">
<attribute id="Category">Equity Income</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">0.8</attribute>
<attribute id="One Year Return">15.2</attribute>
<attribute id="Three Year Return">26.5</attribute>
<attribute id="Load">5.8</attribute>
<attribute id="Minimum Investment">500</attribute>
</product>

<product id="51" name="Fidelity Japan Small Co">
<attribute id="Category">International Stock</attribute>
<attribute id="Morningstar Rating">2</attribute>
<attribute id="Morningstar Risk">1.6</attribute>
<attribute id="One Year Return">83.4</attribute>
<attribute id="Three Year Return">-0.8</attribute>
<attribute id="Load">3.0</attribute>
<attribute id="Minimum Investment">2500</attribute>
</product>
<product id="52" name="Matthews Korea 1">
<attribute id="Category">International Stock</attribute>
<attribute id="Morningstar Rating">1</attribute>
<attribute id="Morningstar Risk">3.2</attribute>
<attribute id="One Year Return">41.6</attribute>
<attribute id="Three Year Return">-19.2</attribute>
<attribute id="Load">0.0</attribute>
<attribute id="Minimum Investment">1000</attribute>
</product>
<product id="81" name="Alger Balanced B">
<attribute id="Category">Balanced</attribute>
<attribute id="Morningstar Rating">3</attribute>
<attribute id="Morningstar Risk">0.7</attribute>
<attribute id="One Year Return">30.7</attribute>
<attribute id="Three Year Return">21.2</attribute>
<attribute id="Load">5.0</attribute>
<attribute id="Minimum Investment">0</attribute>
</product>
<product id="82" name="Idex JCC Balanced A">
<attribute id="Category">Balanced</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">0.5</attribute>

<attribute id="One Year Return">29.5</attribute>
<attribute id="Three Year Return">24.6</attribute>
<attribute id="Load">5.5</attribute>
<attribute id="Minimum Investment">500</attribute>
</product>
<product id="83" name="Janus Balanced">
<attribute id="Category">Balanced</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">0.5</attribute>
<attribute id="One Year Return">28.2</attribute>
<attribute id="Three Year Return">24.3</attribute>
<attribute id="Load">0.0</attribute>
<attribute id="Minimum Investment">2500</attribute>
</product>
<product id="91" name="Calvert Income A">
<attribute id="Category">Corporate Bond</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">0.9</attribute>
<attribute id="One Year Return">11.4</attribute>
<attribute id="Three Year Return">10.0</attribute>
<attribute id="Load">3.8</attribute>
<attribute id="Minimum Investment">2000</attribute>
</product>
<product id="92" name="Crabbe Huson Contrarian Income A">
<attribute id="Category">Corporate Bond</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">1</attribute>
<attribute id="One Year Return">8.2</attribute>
<attribute id="Three Year Return">8.7</attribute>
<attribute id="Load">0.0</attribute>
<attribute id="Minimum Investment">2000</attribute>

```
</product>
<product id="111"
name="CitiFunds National Tax-Free Income A">
<attribute id="Category">Municipal Bond</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">1.1</attribute>
<attribute id="One Year Return">8.3</attribute>
<attribute id="Three Year Return">0.3</attribute>
<attribute id="Load">4.5</attribute>
<attribute id="Minimum Investment">1000</attribute>
</product>
<product id="112"
name="Delaware National High Yield Municpals A">
<attribute id="Category">Municipal Bond</attribute>
<attribute id="Morningstar Rating">5</attribute>
<attribute id="Morningstar Risk">0.2</attribute>
<attribute id="One Year Return">6.6</attribute>
<attribute id="Three Year Return">8.2</attribute>
<attribute id="Load">3.8</attribute>
<attribute id="Minimum Investment">1000</attribute>
</product>
<product id="118" name="Strong Municipal Bond">
<attribute id="Category">Municipal Bond</attribute>
<attribute id="Morningstar Rating">4</attribute>
<attribute id="Morningstar Risk">0.7</attribute>
<attribute id="One Year Return">6.2</attribute>
<attribute id="Three Year Return">8.4</attribute>
<attribute id="Load">0.0</attribute>
<attribute id="Minimum Investment">2500</attribute>
</product>
</catalog>
```

</SETCATALOG>

</BATCHSET>

Use the BATCHSET tag to group commands sent into Precision Choice. Use the SETSTUDY command to indicate to Precision Choice that a study is being loaded. This command contains two parameters: CID and NAME. CID is the command identification used to associate a command response with a command. Precision Choice returns the CID with the response to this command so that the application is able to match up the response to the command with this parameter. The parameter NAME is the key of the study. Precision Choice maintains a persistent copy of this study and subsequent interviews can access it by referring to the NAME. Precision Choice allows only one active study per NAME. The study elements are then contained inside the SETSTUDY command.

The parameters of the SETCATALOG command are similar to those of the SETSTUDY command. The parameters perform the same functions for the catalog that the parameters of the SETSTUDY command do for the study. There can only be one active catalog per catalog name.

Precision Choice loads the study and the catalog and makes them available for future interviews. If the commands are processed successfully, Precision Choice responds with the following XML document.

<BATCHSET>

<CMDRESPONSE CID="1" STATUS="OK"/>

<CMDRESPONSE CID="2" STATUS="OK"/>

</BATCHSET>

A CMDRESPONSE contains two parameters. The CID contains the value that the application sent into Precision Choice, and the STATUS indicates if there were any problems. In this example, both commands were successful. If there were a problem, the STATUS would be ERR and the CMDRESPONSE would contain a MESSAGE element identifying the error.

Once a study and catalog have been loaded into Precision Choice, an interview can begin. To start an interview, the STARTINTERVIEW command must be sent to Precision Choice. It is a batch command and must be contained inside of a BATCH

element. All interview processing commands are batch commands. Multiple interview commands for the same interview can be batched up and sent as one transaction into Precision Choice. The list below identifies the interview commands:

- STARTINTERVIEW
- GETNEXTQUESTIONSET
- SETRESPONSES
- GETSCORES
- GETRECOMMENDATIONS
- ENDINTERVIEW

The sample interview above begins as follows:

First, Precision Choice needs to know how this end-user feels about the various mutual fund categories. Category is a non-a priori attribute, and, therefore, Precision Choice asks a ratings question to learn about the end-user's preferences as related to category. This question might look like the sample in the screenshot below.

To start the interview and get the first set of questions, the following commands must be sent into Precision Choice.

```
<BATCHSET>
<BATCH UID="user1" >
<STARTINTERVIEW CID="3"
STUDY="mutualfunds"
CATALOG="mutualfunds">
</STARTINTERVIEW>
<GETNEXTQUESTIONSET CID="4" MAX="10"/>
</BATCH>
</BATCHSET>
```

The BATCHSET element contains the BATCH of commands to process for an interview. The BATCH element has a UID parameter to allow storage of a user identification. Precision Choice stores the UID with the results of the interview for later analysis. It is required, but if the user specific data will not be collected, the value passed into Precision Choice can be the same for every batch. With this approach, you

will not be able to trace Precision Choice recommendations back to a specific user, but you will still be able to analyze the results of each interview and aggregate the results.

The first command in the batch is the STARTINTERVIEW command. This command, the first command for an interview, indicates which previously loaded study and catalog to use for this interview. The CID parameter performs the same function as mentioned above. The next two parameters point to the study and catalog to use for this interview. Their values should match the NAME attribute from the appropriate SETSTUDY and SETCATALOG commands.

The next command causes Precision Choice to return the first set of questions. For this interview, the questions returned contain one ranking question for the attribute category. The parameter MAX tells Precision Choice how many questions to provide in the question set returned to your application. This example indicates that the application could accept and process ten questions. However, this study only needs one ratings question, so Precision Choice only returns one question. The response from Precision Choice is the following:

```
<BATCHSET><BATCH UID="user1"
INTERVIEWID="user1:947262866239">
<CMDRESPONSE CID="3" STATUS="OK" />
<CMDRESPONSE CID="4" STATUS="OK">
<QUESTIONSET>
<RATINGQUESTION ID="0">
<ATTRIBUTE ID="0">
<LEVEL ID="0" VALUE="Aggressive Growth" />
<LEVEL ID="1" VALUE="Growth" />
<LEVEL ID="2" VALUE="Small Company" />
<LEVEL ID="3" VALUE="International Stock" />
<LEVEL ID="4" VALUE="Balanced" />
<LEVEL ID="5" VALUE="Equity Income" />
<LEVEL ID="6" VALUE="Corporate Bond" />
<LEVEL ID="7" VALUE="Municipal Bond" /></ATTRIBUTE>
<DEFAULTQUESTION></DEFAULTQUESTION>
```

</RATINGQUESTION>
</QUESTIONSET>
</CMDRESPONSE>
</BATCH></BATCHSET>

Note that the responses to the input commands are contained in a BATCHSET element. All responses are like this. Since batch commands were sent into Precision Choice, all responses to the commands are contained in a batch command. The UID that was sent in is returned, and a new INTERVIEWID parameter is also returned. Precision Choice uses the INTERVIEWID to identify which interview the commands are for. All future batches for this interview must contain this value.

The STARTINTERVIEW command (CID="3") was successfully completed. The GETNEXTQUESTIONSET (CID="4") was also completed successfully: it contains the question set that Precision Choice generated.

The only question type contained in the question set is a ratings question, and there is only one ratings question. The attribute Category is the only non-a priori attribute defined in the study; it is the only attribute that Precision Choice needs to ask the user to rate. All other attributes have assumed ratings since they are a priori.

Both the ID of the attribute and the NAME of the attribute are returned. All levels defined in the study are returned with both the ID and VALUE. No DEFAULTQUESTION was defined in the study, so this element is empty (the application would use this information to present this question to the user and gather the answers, if it had been provided). To send the answer into Precision Choice and to get the next set of questions, use the following command.

<BATCHSET>
<BATCH UID="user1" INTERVIEWID="user1:947259311241">
<SETRESPONSES CID="4">
<RATINGRESPONSE ID= "0" >
<LEVEL ID= "0" RATING= "5" />
<LEVEL ID= "1" RATING= "4" />
<LEVEL ID= "2" RATING= "5" />
<LEVEL ID= "3" RATING= "4" />

<LEVEL ID="4" RATING="3" />
<LEVEL ID="5" RATING="2" />
<LEVEL ID="6" RATING="1" />
<LEVEL ID="7" RATING="1" />
</RATINGRESPONSE>
</SETRESPONSES>
<GETNEXTQUESTIONSET CID="5" MAX="10"/>
</BATCH>
</BATCHSET>

The SETRESPONSES command sets the responses to questions. In the example, there is one ratings question. The RATINGSRESPONSE indicates an answer for a ratings question, and the ID indicates the question. This ID should match up to the ID returned in the RATINGQUESTION tag that provided the question. For a ratings response, there must be a rating for each level returned in the RATINGQUESTION element. To indicate the level for an answer, use the level's ID returned in the RATINGQUESTION element. The RATING parameter in the level command indicates the user's rating of the level. The RATING must be a whole number between 1 and 5, with 5 being the most desirable rating. In this example, the end-user strongly prefers the Aggressive Growth and Small Company levels. This end-user prefers the levels Growth and International Stock slightly less, is ambivalent about Balanced, and does not want Equity Income, Corporate Bond, or Municipal Bond funds.

Note carefully that this batch also includes GETNEXTQUESTIONSET with a CID of 5. Precision Choice returns the following response:

<BATCHSET>
<BATCH UID="user1" INTERVIEWID="user1:947252113784">
<CMDRESPONSE CID="4" STATUS="OK" />
<CMDRESPONSE CID="5" STATUS="OK"><QUESTIONSET>
<IMPORTANCEQUESTION ID="1">
<ATTRIBUTE
ID="0"
NAME="Category"

```
BEST="Aggressive Growth"
WORST="Municipal Bond" />
<DEFAULTQUESTION></DEFAULTQUESTION>
</IMPORTANCEQUESTION>
<IMPORTANCEQUESTION ID="2">
<ATTRIBUTE
ID="1"
NAME="Morningstar Rating"
BEST="5"
WORST="1"/>
<DEFAULTQUESTION></DEFAULTQUESTION>
</IMPORTANCEQUESTION>
<IMPORTANCEQUESTION ID="3">
<ATTRIBUTE
ID="2"
NAME="Morningstar Risk"
BEST="0.5"
WORST="1.5" />
<DEFAULTQUESTION></DEFAULTQUESTION>
</IMPORTANCEQUESTION>
<IMPORTANCEQUESTION ID="4">
<ATTRIBUTE
ID="3"
NAME="Three Year Return"
BEST="30"
WORST="0" />
<DEFAULTQUESTION></DEFAULTQUESTION>
</IMPORTANCEQUESTION>
<IMPORTANCEQUESTION ID="5">
<ATTRIBUTE
ID="4"
```

NAME="Load"
BEST="0"
WORST="6" />
<DEFAULTQUESTION></DEFAULTQUESTION>
</IMPORTANCEQUESTION></QUESTIONSET>
</CMDRESPONSE>
</BATCH>
</BATCHSET>

The CMDRESPONSE with a CID value of 4 indicates that the SETRESPONSE command was processed successfully. The next response contains the importance questions and is a response to the GETNEXTQUESTIONSET command. Since a maximum of ten questions was specified in the GETNEXTQUESTIONSET, Precision Choice returns all five importance questions. This interaction matches up with the sample interview above at the point below:

In this study, there are no remaining non-a priori attributes. The remaining attributes (Morningstar Rating, Morningstar Risk, Three Year Return, and Load) are all a priori. This is why Precision Choice does not ask ratings questions for the remaining attributes.

Precision Choice then asks importance questions to learn what importance the end-user places on each attribute. In this study, the next five questions would be importance questions. Precision Choice generates these five questions at one time because it knows the elements necessary to generate each one in advance.

Each importance question contains an ATTRIBUTE to identify which study attribute the question is about. The element also contains parameters to indicate the best and worst levels of the attribute, according to the end-user's ranking. If the attribute is a priori, the level ranking is determined by the order the levels are defined in the study. No importance default question was defined in the study, so DEFAULTQUESTION is empty. To answer all importance questions and to get the next set of questions from Precision Choice, use the following syntax:

<BATCHSET>
<BATCH UID="user1" INTERVIEWID="user1:947252113784">

```
<SETRESPONSES CID="6">
<IMPORTANCERESPONSE ID= "1" >
<ATTRIBUTE ID= "0" RATING= "3" />
</IMPORTANCERESPONSE>

<IMPORTANCERESPONSE ID= "2" >
<ATTRIBUTE ID= "0" RATING= "4" />
</IMPORTANCERESPONSE>

<IMPORTANCERESPONSE ID= "3" >
<ATTRIBUTE ID= "0" RATING= "2" />
</IMPORTANCERESPONSE>

<IMPORTANCERESPONSE ID= "4" >
<ATTRIBUTE ID= "0" RATING= "4" />
</IMPORTANCERESPONSE>

<IMPORTANCERESPONSE ID= "5" >
<ATTRIBUTE ID= "0" RATING= "3" />
</IMPORTANCERESPONSE>
</SETRESPONSES>
<GETNEXTQUESTIONSET CID="7" MAX="10"/>
</BATCH>
</BATCHSET>
```

In this BATCHSET, the SETRESPONSES command contains IMPORTANCERESPONSE elements, as opposed to the RATINGSRESPONSE in the previous SETRESPONSES command. This indicates answers for importance questions. The ID parameter must match up to the ID returned previously in the IMPORTANCEQUESTION element. This tells Precision Choice which importance question is being answered. Each importance response contains an ATTRIBUTE element to indicate the attribute that the answer is for. This information comes from the

```
<SETRESPONSES CID="6">
<IMPORTANCERESPONSE ID= "1" >
<ATTRIBUTE ID= "0" RATING= "3" />
</IMPORTANCERESPONSE>

<IMPORTANCERESPONSE ID= "2" >
<ATTRIBUTE ID= "0" RATING= "4" />
</IMPORTANCERESPONSE>

<IMPORTANCERESPONSE ID= "3" >
<ATTRIBUTE ID= "0" RATING= "2" />
</IMPORTANCERESPONSE>

<IMPORTANCERESPONSE ID= "4" >
<ATTRIBUTE ID= "0" RATING= "4" />
</IMPORTANCERESPONSE>

<IMPORTANCERESPONSE ID= "5" >
<ATTRIBUTE ID= "0" RATING= "3" />
</IMPORTANCERESPONSE>
</SETRESPONSES>
<GETNEXTQUESTIONSET CID="7" MAX="10"/>
</BATCH>
</BATCHSET>
```

In this BATCHSET, the SETRESPONSES command contains IMPORTANCERESPONSE elements, as opposed to the RATINGSRESPONSE in the previous SETRESPONSES command. This indicates answers for importance questions. The ID parameter must match up to the ID returned previously in the IMPORTANCEQUESTION element. This tells Precision Choice which importance question is being answered. Each importance response contains an ATTRIBUTE element to indicate the attribute that the answer is for. This information comes from the

ATTRIBUTE tag contained in the IMPORTANCEQUESTION element previously returned. The rating should be a whole number between 1 and 4, inclusive. Four represents the highest importance to the user. In the sample command shown above, the user places the highest importance on the attributes Morningstar Rating and Three-Year Return. The user rates the attributes Category and Load slightly less important and assigns lower importance to Morningstar Risk. This BATCHSET also contains the GETNEXTQUESTIONSET command to instruct Precision Choice to send the next set of questions. Precision Choice now moves to the following section of the sample interview:

Precision Choice then moves into the pairs questions. Precision Choice analyzes what it knows of the end-user's preferences to this point and uses that information to build the toughest trade-off question it can. The first pairs question contains only two attributes

The response to the previously discussed BATCHSET is as follows:

```
<BATCHSET>
<BATCH UID="user1" INTERVIEWID="user1:947252113784">
<CMDRESPONSE CID="6" STATUS="OK" />
<CMDRESPONSE CID="7" STATUS="OK">
<QUESTIONSET><PAIRSQUESTION ID="6">
<SAMPLE SIDE="low">
<ATTRIBUTE ID="1" NAME="Morningstar Rating">
<LEVEL VALUE="5" />
</ATTRIBUTE>
<ATTRIBUTE ID="3" NAME="Three Year Return">
<LEVEL VALUE="20"/>
</ATTRIBUTE>
</SAMPLE>
<SAMPLE SIDE="high">
<ATTRIBUTE
ID="1"
NAME="Morningstar Rating">
```

<LEVEL VALUE="4" />
</ATTRIBUTE>
<ATTRIBUTE
ID="3"
NAME="Three Year Return">
<LEVEL VALUE="30" >
</ATTRIBUTE>
</SAMPLE>
<DEFAULTQUESTION></DEFAULTQUESTION>
</PAIRSQUESTION>
</QUESTIONSET>
</CMDRESPONSE>
</BATCH>
</BATCHSET>

The SETRESPONSES command was completed successfully, and Precision Choice returns the next question set. This question set contains one pairs question. Since Precision Choice dynamically generates pairs questions based upon the user's previous questions, only one is available at a time. The PAIRS QUESTION element contains two sample products with two attributes each. The end-user's response is on a scale of 1 to 9. The closer the end-user's response is to 1, the more he or she prefers the "low" sample. The closer the end-user's response is to 9, the more he or she prefers the "high" sample. A choice of 5 indicates that the end-user has no preference between the two sample products.

The next interaction with Precision Choice provides an answer to this question and gets the next question set.

<BATCHSET>
<BATCH UID="user1" INTERVIEWID="user1:947252113784">
<SETRESPONSES CID="8">
<PAIRSRESPONSE ID= "6" RATING = "7"/>
</SETRESPONSES>
<GETNEXTQUESTIONSET CID="9" MAX="10"/>

```
</BATCH>
</BATCHSET>
```

This BATCHSET tells Precision Choice that this end-user prefers the "high" sample more than the "low" sample. The response to this batchset would look like the following:

```
<BATCHSET>
<BATCH UID="user1" INTERVIEWID="user1:947252113784">
<CMDRESPONSE CID="8" STATUS="OK" />
<CMDRESPONSE CID="9" STATUS="OK">
<QUESTIONSET><PAIRSQUESTION ID="7">
<SAMPLE SIDE="low">
<ATTRIBUTE ID="0" NAME="Category">
<LEVEL VALUE="Aggressive Growth" /></ATTRIBUTE>
<ATTRIBUTE ID="4" NAME="Load">
<LEVEL VALUE="2" /></ATTRIBUTE>
</SAMPLE>
<SAMPLE SIDE="high">
<ATTRIBUTE ID="0" NAME="Category">
<LEVEL VALUE="Small Company" /></ATTRIBUTE>
<ATTRIBUTE ID="4" NAME="Load">
<LEVEL VALUE="0" /></ATTRIBUTE>
</SAMPLE>
<DEFAULTQUESTION></DEFAULTQUESTION>
</PAIRSQUESTION></QUESTIONSET>
</CMDRESPONSE></BATCH></BATCHSET>
```

The answer to the previous question was accepted, and the next pairs question was generated. Precision Choice is now at the following point in the sample interview above:

After Precision Choice receives the end-user's response to the first pairs question, it adjusts the preference profile to incorporate what it has learned. Precision

Choice again builds the toughest trade-off question that it can. The second pairs question for the user contains only two attributes.

To indicate to Precision Choice that the end-user prefers the "high" sample and to get the next question set, send the following BATCHSET:

```
<BATCHSET>
<BATCH UID="user1" INTERVIEWID="user1:947252113784">
<SETRESPONSES CID="9">
<PAIRSRESPONSE ID= "7"  RATING = "9"/>
</SETRESPONSES>
<GETNEXTQUESTIONSET CID="10" MAX="10"/>
</BATCH>
</BATCHSET>
```

Precision Choice's response to this BATCHSET is the following:

```
<BATCHSET>
<BATCH
UID="user1"
INTERVIEWID="user1:947259311241">
<CMDRESPONSE CID="6" STATUS="OK" />
<CMDRESPONSE CID="7" STATUS="OK">
<QUESTIONSET>
<PAIRSQUESTION ID="10">
<SAMPLE SIDE="low">
<ATTRIBUTE
ID="3"
NAME="Three Year Return">
<LEVEL VALUE="10" />
</ATTRIBUTE>
<ATTRIBUTE
ID="0"
NAME="Category">
<LEVEL VALUE="Aggressive Growth" />
```

```
</ATTRIBUTE>
<ATTRIBUTE
ID="2"
NAME="Morningstar Risk">
<LEVEL VALUE="0.5" />
</ATTRIBUTE>
</SAMPLE>
<SAMPLE SIDE="high">
<ATTRIBUTE
ID="3"
NAME="Three Year Return">
<LEVEL VALUE="30" />
</ATTRIBUTE>
<ATTRIBUTE
ID="0"
NAME="Category">
<LEVEL VALUE="Growth" />
</ATTRIBUTE>
<ATTRIBUTE
ID="2"
NAME="Morningstar Risk">
<LEVEL VALUE="1.0" />
</ATTRIBUTE>
</SAMPLE>
<DEFAULTQUESTION></DEFAULTQUESTION>
</PAIRSQUESTION>
</QUESTIONSET>
</CMDRESPONSE>
</BATCH></BATCHSET>
```

The answer to the previous question was accepted and the next pairs question was generated. Precision Choice has reached the following point in the sample interview above:

Precision Choice generates the third pairs question after it receives the response to the second pairs question and incorporates it into the preference profile. At this point, Precision Choice presents pairs questions with three attributes each.

To indicate to Precision Choice that the end-user prefers the "high" sample, use the following BATCHSET:

```
<BATCHSET>
<BATCH UID="user1" INTERVIEWID="user1:947259311241">
SETRESPONSES CID="6">
<PAIRSRESPONSE ID="10" RATING="7"/>
</SETRESPONSES>
<GETNEXTQUESTIONSET CID="7" MAX="10"/>
</BATCH>
</BATCHSET>
```

Precision Choice responds to this BATCHSET as follows:

```
<BATCHSET>
<BATCH
UID="user1"
INTERVIEWID="user1:947259311241">
<CMDRESPONSE CID="6" STATUS="OK" />
<CMDRESPONSE CID="7" STATUS="OK">
<QUESTIONSET>
<PAIRSQUESTION ID="11">
<SAMPLE SIDE="low">
<ATTRIBUTE
ID="3"
NAME="Three Year Return">
<LEVEL VALUE="30" /></ATTRIBUTE>
<ATTRIBUTE
```

ID="4"
NAME="Load">
<LEVEL VALUE="6" /></ATTRIBUTE>
<ATTRIBUTE
ID="2"
NAME="Morningstar Risk">
<LEVEL VALUE="1.25" /></ATTRIBUTE>
</SAMPLE>
<SAMPLE SIDE="high">
<ATTRIBUTE
ID="3"
NAME="Three Year Return">
<LEVEL VALUE="0" /></ATTRIBUTE>
<ATTRIBUTE
ID="4"
NAME="Load">
<LEVEL VALUE="0" /></ATTRIBUTE>
<ATTRIBUTE
ID="2"
NAME="Morningstar Risk">
<LEVEL VALUE="0.75" /></ATTRIBUTE>
</SAMPLE>
<DEFAULTQUESTION></DEFAULTQUESTION>
</PAIRSQUESTION></QUESTIONSET>
</CMDRESPONSE>
</BATCH></BATCHSET>

The answer to the previous question is accepted and the next pairs question is generated. Precision Choice has reached the following point from the sample interview above:

The fourth pairs question also has three attributes. The questions are becoming harder because Precision Choice is honing in precisely on the end-user's preferences.

The following BATCHSET indicates to Precision Choice that the end-user prefers the "low" sample:

```
<BATCHSET>
<BATCH UID="user1" INTERVIEWID="user1:947259311241">
<SETRESPONSES CID="6">
<PAIRSRESPONSE ID="11" RATING="3"/>
</SETRESPONSES>
<GETNEXTQUESTIONSET CID="7" MAX="10"/>
</BATCH>
</BATCHSET>
```

Precision Choice responds with the following:

```
<BATCHSET>
<BATCH
UID="user1"
INTERVIEWID="user1:947259311241">
<CMDRESPONSE CID="6" STATUS="OK" />
<CMDRESPONSE CID="7" STATUS="OK">
<QUESTIONSET>
<PAIRSQUESTION ID="12">
<SAMPLE SIDE="low">
<ATTRIBUTE
ID="1"
NAME="Morningstar Rating">
<LEVEL VALUE="3" /></ATTRIBUTE>
<ATTRIBUTE
ID="3"
NAME="Three Year Return">
<LEVEL VALUE="20" /></ATTRIBUTE>
<ATTRIBUTE
ID="0"
NAME="Category">
```

<LEVEL VALUE="International Stock" /></ATTRIBUTE>
<ATTRIBUTE
ID="4"
NAME="Load">
<LEVEL VALUE="2" /></ATTRIBUTE>
</SAMPLE>
<SAMPLE SIDE="high">
<ATTRIBUTE
ID="1"
NAME="Morningstar Rating">
<LEVEL VALUE="4" /></ATTRIBUTE>
<ATTRIBUTE
ID="3"
NAME="Three Year Return">
<LEVEL VALUE="10" /></ATTRIBUTE>
<ATTRIBUTE
ID="0"
NAME="Category">
<LEVEL VALUE="Small Company" /></ATTRIBUTE>
<ATTRIBUTE
ID="4"
NAME="Load">
<LEVEL VALUE="4" /></ATTRIBUTE>
</SAMPLE>
<DEFAULTQUESTION></DEFAULTQUESTION>
</PAIRSQUESTION>
</QUESTIONSET>
</CMDRESPONSE>
</BATCH></BATCHSET>

The answer to the previous question is accepted and the next pairs question is generated. Precision Choice has reached the following point from the sample interview above:

Precision Choice then generates the seventh pairs question, the only pairs question with four attributes.

Send the following BATCHSET to indicate to Precision Choice that the end-user prefers the "low" sample:

```
<BATCHSET>
<BATCH UID="user1" INTERVIEWID="user1:947259311241">
<SETRESPONSES CID="6">
<PAIRSRESPONSE ID="12" RATING="3"/>
</SETRESPONSES>
<GETNEXTQUESTIONSET CID="7" MAX="10"/>
</BATCH>
</BATCHSET>
```

Precision Choice responds with the following:

```
<BATCHSET>
<BATCH
UID="user1"
INTERVIEWID="user1:947259311241">
<CMDRESPONSE CID="6" STATUS="OK" />
<CMDRESPONSE CID="7" STATUS="OK">
<QUESTIONSET />
</CMDRESPONSE>
</BATCH></BATCHSET>
```

In this response, Precision Choice indicates that it accepted the end-user's answer to the last Pairs question and that there are no more questions to generate. When Precision Choice sends an empty QUESTIONSET element, as in the sample code above (<QUESTIONSET />), there are no more questions in the interview.

Precision Choice would now ask calibration questions, if it were configured to do so. Calibration questions are not asked by default, and therefore are not included in this example.

Precision Choice's questioning process is complete, and Precision Choice is ready to provide product recommendations to the user. Precision Choice calculates the utility scores and then compares this information to the products in the catalog. The products are then ranked and returned. Before getting the product recommendations from Precision Choice, you must get the scores. To calculate the utilities and return the first page of products, send the following BATCHSET into Precision Choice:

```
<BATCHSET>
<BATCH UID="user1" INTERVIEWID="user1:947259311241">
<GETSCORES CID="1"/>
<GETRECOMMENDATIONS CID="2" MAX="5"/>
</BATCH>
</BATCHSET>
```

The GETSCORES command is an empty tag that only specifies a command id. The GETRECOMMENDATIONS command also defines a command id and the maximum number of products to return. Precision Choice responds to this batch set with the following response.

```
<BATCHSET>
<BATCH
UID="user1"
INTERVIEWID="user1:947259311241">
<CMDRESPONSE CID="1" STATUS="OK">
<SCORES>
<ATTRIBUTE
NAME="Category"
RELATIVEIMPORTANCE="0.18824428756238512">
<LEVEL
ID="7"
NAME=""
```

```
VALUE="Municipal Bond">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="0.0" />
</LEVEL>
<LEVEL
ID="6"
NAME=""
VALUE="Corporate Bond">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="0.0" />
</LEVEL>
<LEVEL
ID="5"
NAME=""
VALUE="Equity Income">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="8.0" />
</LEVEL>
<LEVEL
ID="4"
NAME=""
VALUE="Balanced">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="17.0" />
</LEVEL>
<LEVEL
ID="3"
```

```
NAME=""
VALUE="International Stock">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="37.0" />
</LEVEL>
<LEVEL
ID="2"
NAME=""
VALUE="Small Company">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="24.0" />
</LEVEL>
<LEVEL
ID="1"
NAME=""
VALUE="Growth">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="25.0" />
</LEVEL>
<LEVEL
ID="0"
NAME=""
VALUE="Aggressive Growth">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="37.0" />
</LEVEL>
</ATTRIBUTE>
```

```
<ATTRIBUTE
NAME="Morningstar Rating"
RELATIVEIMPORTANCE="0.2509938521942569">
<LEVEL
ID="4"
NAME=""
VALUE="1">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="17.0" />
</LEVEL>
<LEVEL
ID="3"
NAME=""
VALUE="2">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="37.0" />
</LEVEL>
<LEVEL
ID="2"
NAME=""
VALUE="3">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="24.0" />
</LEVEL>
<LEVEL
ID="1" NAME=""
VALUE="4">
<UTILITY
```

```
NAME="FinalScaledUtilities"
VALUE="25.0" />
</LEVEL>
<LEVEL
ID="0"
NAME=""
VALUE="5">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="37.0" />
</LEVEL>
</ATTRIBUTE>
<ATTRIBUTE
NAME="Morningstar Risk"
RELATIVEIMPORTANCE="0.09292788744402652">
<LEVEL
ID="4"
NAME=""
VALUE="1.5">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="17.0" />
</LEVEL>
<LEVEL
ID="3"
NAME=""
VALUE="1.25">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="37.0" />
</LEVEL>
```

```
<LEVEL
ID="2"
NAME=""
VALUE="1.0">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="24.0" />
</LEVEL>
<LEVEL
ID="1"
NAME=""
VALUE="0.75">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="25.0" />
</LEVEL>
<LEVEL
ID="0"
NAME=""
VALUE="0.5">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="37.0" />
</LEVEL>
</ATTRIBUTE>
<ATTRIBUTE
NAME="Three Year Return"
RELATIVEIMPORTANCE="0.3048983003353743">
<LEVEL
ID="3"
NAME=""
```

```
VALUE="0">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="37.0" />
</LEVEL>
<LEVEL
ID="2"
NAME=""
VALUE="10">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="24.0" />
</LEVEL>
<LEVEL
ID="1"
NAME=""
VALUE="20">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="25.0" />
</LEVEL>
<LEVEL
ID="0"
NAME=""
VALUE="30">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="37.0" />
</LEVEL>
</ATTRIBUTE>
<ATTRIBUTE
```

```
NAME="Load"
RELATIVEIMPORTANCE="0.1629356724639573">
<LEVEL
ID="3"
NAME=""
VALUE="6">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="37.0" />
</LEVEL>
<LEVEL
ID="2"
NAME=""
VALUE="4">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="24.0" />
</LEVEL>
<LEVEL
ID="1"
NAME=""
VALUE="2">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="25.0" />
</LEVEL>
<LEVEL
ID="0"
NAME=""
VALUE="0">
<UTILITY
```

```
NAME="FinalScaledUtilities"
VALUE="37.0" />
</LEVEL>
</ATTRIBUTE
></SCORES>
</CMDRESPONSE>
<CMDRESPONSE CID="2" STATUS="OK">
<MAXPRODUCTS VALUE="18" />
<PRODUCTS>
<PRODUCT
ID="1"
NAME="Janus Olympus"
FIT="100.0"
RANK="1">
<ATTRIBUTE
NAME="Three Year Return"
VALUE="39.0"
FIT="100.0"
RELATIVEIMPORTANCE="30.0" />
<ATTRIBUTE
NAME="Minimum Investment"
VALUE="2500"
FIT="-1.0"
RELATIVEIMPORTANCE="-1.0" />
<ATTRIBUTE
NAME="Load"
VALUE="0.0"
FIT="100.0"
RELATIVEIMPORTANCE="16.0" />
<ATTRIBUTE
NAME="Category"
```

```
VALUE="Aggressive Growth"
FIT="100.0"
RELATIVEIMPORTANCE="19.0" />
<ATTRIBUTE
NAME="Morningstar Risk"
VALUE="1.2"
FIT="56.0"
RELATIVEIMPORTANCE="9.0" />
<ATTRIBUTE
NAME="Morningstar Rating"
VALUE="5"
FIT="100.0"
RELATIVEIMPORTANCE="25.0" />
<ATTRIBUTE
NAME="One Year Return"
VALUE="66.5"
FIT="-1.0"
RELATIVEIMPORTANCE="-1.0" />
</PRODUCT>
<PRODUCT
ID="11"
NAME="Janus Twenty"
FIT="96.0"
RANK="2">
<ATTRIBUTE
NAME="Three Year Return"
VALUE="47.9"
FIT="100.0"
RELATIVEIMPORTANCE="30.0" />
<ATTRIBUTE
NAME="Minimum Investment"
```

```
VALUE="2500"
FIT="-1.0"
RELATIVEIMPORTANCE="-1.0" />
<ATTRIBUTE
NAME="Load"
VALUE="0.0"
FIT="100.0"
RELATIVEIMPORTANCE="16.0" />
<ATTRIBUTE
NAME="Category"
VALUE="Growth"
FIT="68.0"
RELATIVEIMPORTANCE="19.0" />
<ATTRIBUTE
NAME="Morningstar Risk"
VALUE="0.9"
FIT="78.0"
RELATIVEIMPORTANCE="9.0" />
<ATTRIBUTE
NAME="Morningstar Rating"
VALUE="5"
FIT="100.0"
RELATIVEIMPORTANCE="25.0" />
<ATTRIBUTE
NAME="One Year Return"
VALUE="79.9"
FIT="-1.0"
RELATIVEIMPORTANCE="-1.0" />
</PRODUCT>
<PRODUCT
ID="32"
```

```
NAME="Robertson Stephens Emerging Growth A"
FIT="82.0"
RANK="3">
<ATTRIBUTE
NAME="Three Year Return"
VALUE="31.6"
FIT="100.0"
RELATIVEIMPORTANCE="30.0" />
<ATTRIBUTE
NAME="Minimum Investment"
VALUE="5000"
FIT="-1.0"
RELATIVEIMPORTANCE="-1.0" />
<ATTRIBUTE
NAME="Load"
VALUE="0.0"
FIT="100.0"
RELATIVEIMPORTANCE="16.0" />
<ATTRIBUTE
NAME="Category"
VALUE="Small Company"
FIT="65.0"
RELATIVEIMPORTANCE="19.0" />
<ATTRIBUTE
NAME="Morningstar Risk"
VALUE="1.7"
FIT="0.0"
RELATIVEIMPORTANCE="9.0" />
<ATTRIBUTE
NAME="Morningstar Rating"
VALUE="4"
```

```
FIT="76.0"
RELATIVEIMPORTANCE="25.0" />
<ATTRIBUTE
NAME="One Year Return"
VALUE="38.5"
FIT="-1.0"
RELATIVEIMPORTANCE="-1.0" />
</PRODUCT>
<PRODUCT
ID="83"
NAME="Janus Balanced"
FIT="78.0"
RANK="4">
<ATTRIBUTE
NAME="Three Year Return"
VALUE="24.3"
FIT="72.0"
RELATIVEIMPORTANCE="30.0" />
<ATTRIBUTE
NAME="Minimum Investment"
VALUE="2500"
FIT="-1.0"
RELATIVEIMPORTANCE="-1.0" />
<ATTRIBUTE
NAME="Load"
VALUE="0.0"
FIT="100.0"
RELATIVEIMPORTANCE="16.0" />
<ATTRIBUTE
NAME="Category"
VALUE="Balanced"
```

```
FIT="46.0"
RELATIVEIMPORTANCE="19.0" />
<ATTRIBUTE
NAME="Morningstar Risk"
VALUE="0.5"
FIT="100.0"
RELATIVEIMPORTANCE="9.0" />
<ATTRIBUTE
NAME="Morningstar Rating"
VALUE="4"
FIT="76.0"
RELATIVEIMPORTANCE="25.0" />
<ATTRIBUTE
NAME="One Year Return"
VALUE="28.2"
FIT="-1.0"
RELATIVEIMPORTANCE="-1.0" />
</PRODUCT>
<PRODUCT
ID="4"
NAME="Alger Capital Appreciation B"
FIT="78.0"
RANK="5">
<ATTRIBUTE
NAME="Three Year Return"
VALUE="28.1"
FIT="90.0"
RELATIVEIMPORTANCE="30.0" />
<ATTRIBUTE
NAME="Minimum Investment"
VALUE="0"
```

```
FIT="-1.0"
RELATIVEIMPORTANCE="-1.0" />
<ATTRIBUTE
NAME="Load"
VALUE="5.0"
FIT="9.0"
RELATIVEIMPORTANCE="16.0" />
<ATTRIBUTE
NAME="Category"
VALUE="Aggressive Growth"
FIT="100.0"
RELATIVEIMPORTANCE="19.0" />
<ATTRIBUTE
NAME="Morningstar Risk"
VALUE="1.4"
FIT="22.0"
RELATIVEIMPORTANCE="9.0" />
<ATTRIBUTE
NAME="Morningstar Rating"
VALUE="5"
FIT="100.0"
RELATIVEIMPORTANCE="25.0" />
<ATTRIBUTE
NAME="One Year Return"
VALUE="47.4"
FIT="-1.0"
RELATIVEIMPORTANCE="-1.0" />
</PRODUCT>
</PRODUCTS>
</CMDRESPONSE>
</BATCH>
```

</BATCHSET>

The first CMDRESPONSE in this BATCHSET indicates that the GETSCORES command was completed successfully. Scores are produced for each attribute and each level defined in the study. The ATTRIBUTE tag displays a new parameter, RELATIVEIMPORTANCE. Every attribute in the study has RELATIVEIMPORTANCE value that is decimal. Each LEVEL of an attribute is returned, and this element contains the calculated utilities for that level. Only the final scaled utility value is contained in the response. For details on how to interpret and use the final scaled utility, see the foregoing discussion.

The second command response contains the recommended products. Since the GETRECOMMENDATIONS command set MAX to five, five products are returned. The total number of products in the catalog is also returned in the MAXPRODUCTS element. This value is used to display to the end-user statements like "Product 1 of 18."The products are contained in a PRODUCTS element. Two new parameters are displayed in the PRODUCTS tag: FIT and RANK. FIT is a number between 0 and 100 that indicates how closely this product matches the end-user's product preferences. Every product in the recommendation set contains this value. The RANK parameter identifies where this product stands in relation to the other products in the catalog. RANK number one is the product that best matches the end-user's preferences. All product attributes, not just the attributes defined in the study, are returned in the PRODUCT element. FIT and RELATIVEIMPORTANCE parameters are added to the ATTRIBUTE tag. The FIT parameter defines how closely the value of the product's attribute matches the end-user's preferences. If this attribute is not defined in the study, FIT is -1. RELATIVEIMPORTANCE defines how important this attribute is to the end-user. It is -1 when this attribute is not defined in the study.

The first time that you call GETRECOMMENDATIONS, Precision Choice stores the user's preferences and top five product recommendations for later analysis. To continue with more product recommendations, you need to send in the GETRECOMMENDATIONS command specifying how many products to return. You can only move down the list.

Once an end-user has finished viewing all of the recommendations that he or she wants to see, the end-user should end the interview. Ending the interview allows Precision Choice to reclaim any resources in use for this interview. To get another set of recommendations and to end the interview, send the following BATCHSET:

```
<BATCHSET>
<BATCH UID="user1" INTERVIEWID="user1:947259311241">
<GETRECOMMENDATIONS CID="2" MAX="1"/>
<ENDINTERVIEW CID="3" />
</BATCH>
</BATCHSET>
```

Precision Choice responds with the following BATCHSET:

```
<BATCHSET>
<BATCH
UID="user1"
INTERVIEWID="user1:947259311241">
<CMDRESPONSE CID="2" STATUS="OK">
<MAXPRODUCTS VALUE="18" />
<PRODUCTS>
<PRODUCT
ID="14"
NAME="WM Growth A"
FIT="72.0"
RANK="6">
<ATTRIBUTE
NAME="Three Year Return"
VALUE="31.8"
FIT="100.0"
RELATIVEIMPORTANCE="30.0" />
<ATTRIBUTE
NAME="Minimum Investment"
VALUE="1000"
```

```
FIT="-1.0"
RELATIVEIMPORTANCE="-1.0" />
<ATTRIBUTE
NAME="Load"
VALUE="5.5"
FIT="6.0"
RELATIVEIMPORTANCE="16.0" />
<ATTRIBUTE
NAME="Category"
VALUE="Growth"
FIT="68.0"
RELATIVEIMPORTANCE="19.0" />
<ATTRIBUTE
NAME="Morningstar Risk"
VALUE="1.2"
FIT="56.0"
RELATIVEIMPORTANCE="9.0" />
<ATTRIBUTE
NAME="Morningstar Rating"
VALUE="4"
FIT="76.0"
RELATIVEIMPORTANCE="25.0" />
<ATTRIBUTE
NAME="One Year Return"
VALUE="63.3"
FIT="-1.0"
RELATIVEIMPORTANCE="-1.0" />
</PRODUCT>
</PRODUCTS>
</CMDRESPONSE>
<CMDRESPONSE CID="3" STATUS="OK" />
```

</BATCH></BATCHSET>

This BATCHSET shows that the second GETTRECOMMENDATIONS command was successfully completed. This time one product is returned, the sixth product in rank, since the command specified MAX of 1. ENDINTERVIEW, the second command, was also completed successfully. The interview is complete, and Precision Choice has reclaimed all resources by this point.

Precision Choice XML Command Language

The forgoing present elements and attributes of the Precision Choice command language. Commands are grouped according to their function: Administration or Interview

The Administration commands are the following:

- SETSTUDY
- SETCATALOG
- GETSTATUS

The Interview commands are the following:

- STARTINTERVIEW
- GETNEXTQUESTIONSET
- SETRESPONSES
- GETSCORES
- GETRECOMMENDATIONS
- ENDINTERVIEW

All Precision Choice commands are defined in a inputcmd.dtd file. All responses from Precision Choice are defined in a outputcmd.dtd file.

Administration Commands

Administration commands are not used within the context of a specific interview. Usually, they are used to provide information for the interview sessions in Precision Choice.

SETSTUDY Command

Use the SETSTUDY command to load a study into Precision Choice. Interviews are based upon studies that must be loaded into Precision Choice before an

interview can reference the study. Normally, the defined studies are loaded when Precision Choice is started, but a study can be loaded as part of an interview by preceding the STARTINTERVIEW command with the SETSTUDY command.

The SETSTUDY command has two formats. One format allows a study file to be loaded into Precision Choice. This format is a tag with only attributes. The other format allows the study XML document to be defined and loaded directly in the SETSTUDY command. This format defines the SETSTUDY command with two attributes and one element that contains the study definition.

Format 1 of the SETSTUDY Command:

```
<SETSTUDY CID="1" NAME="computerfinder">
<study name="computerfinder" version="1.0">
<attribute desc="Brand" unit="text">
<level>Compaq</level>
<level>Dell</level>
<level>Gateway</level>
<level>HP</level>
<level>IBM</level>
</attribute>
<attribute
desc="Price"
unit="currency"
interpolate="yes"
apriori="yes">
<level>1800</level>
<level>2600</level>
<level>3400</level>
<level>4200</level>
<level>5000</level>
</attribute>
</study>
</SETSTUDY>
```

Elements of Format 1 of the SETSTUDY Command

Note: XML is case-sensitive.

SETSTUDY: Tag containing the study definition.

CID: The command identification returned with the response to this command. This value can be used to associate the Precision Choice response with the command that caused it. It is alphanumeric and required. Precision Choice does not use the value, but returns it in the response to the command.

NAME: The name the STARTINTERVIEW command uses to reference the study that you are loading. It is alphanumeric and required.

study Definition

The study definition is used to define a study in Precision Choice. The study provides information Precision Choice needs to perform interviews with users. A study defines features of a particular product that are used to determine end-user preferences. For example, if the product were a laptop computer, attributes might include price, brand, CPU, and hard drive size. Only features that impact end-users' decision-making processes should be defined in the study. The study definition is never sent directly to Precision Choice.

Format of the study Definition

```
<study name="notebookcomputers" version="1.0">
<attribute desc="Brand" unit="text">
<level>Compaq</level>
<level>Dell</level>
<level>Gateway</level>
<level>HP</level>
<level>IBM</level>
</attribute>
<attribute
desc="Price"
unit="currency"
interpolate="yes"
apriori="yes">
```

<level>1800</level>
<level>2600</level>
<level>3400</level>
<level>4200</level>
<level>5000</level>
</attribute>
<defaultquestions>
<defaultquestion
type="rating">Please, rate these attributes
</defaultquestion>
<defaultquestion
type="pairs">Please, choose a sample.
</defaultquestion>
</defaultquestions>
</study>

Elements of the study Definition

Note: XML is case-sensitive.

study: Tag containing elements that define a study.

name: The study's name. This name is for the study administrator's purposes only. Precision Choice does not use it. This is not the same as the NAME element of the SETSTUDY command.

version: The study's version. The version is for the study administrator's purposes only. Precision Choice does not use it.

attribute: Defines an attribute of the study. An attribute is a feature that users would find important about the product. There must be at least three attributes defined in a study.

desc: A textual description of an attribute. This information becomes part of the questions Precision Choice generates and is required for each attribute. A desc is alphanumeric and can contain multiple words.

units: Defines how to display the attribute's levels. The units must be defined in the units.xml file and are required.

interpolate: Tells Precision Choice whether values of the levels can be interpolated. Typically, attributes such as "Brand" are not interpolated; attributes such as "Price" are interpolated. This element is not required. The elements can be either "Yes" or "No." If it is not present, the value is assumed to be "no."

apriori: If the attribute's levels have an assumed choice, the value should be "Yes." An example of this element is "Price." When provided a choice, almost everyone wants the lowest price. The order in which levels are defined in the XML document determines the best and worst values. The first level defined is assumed to be the best choice, and the last level defined is assumed to be the least desirable choice. This element is not required, and the default value is "No."

level: Defines some possible values for an attribute. At least two levels per attribute are required.

defaultquestions: This tag contains the default questions for this study.

defaultquestion: This tag defines a default question to use for each question type that Precision Choice supports: rating, importance, pairs, and calibration. This element is not required. Default questions are returned with the response to GETNEXTQUESTIONSET. The application can decide whether to use the text.

type: This is the value for the type of question: rating, importance, pairs, or calibration.

Format 2 of the SETSTUDY Command

```
<SETSTUDY
CID="1"
NAME="computerfinder"
URL="http://www.onlineinsight.com/study.xml" />
```

Elements of Format 2 of the SETSTUDY Command

Note: XML is case-sensitive.

SETSTUDY: Command tag for setting the study. In this case, it does not contain additional attributes.

CID: The command identification returned with the response to this command. This value is used to associate the Precision Choice response with the command that

caused it. It is alphanumeric and required. Precision Choice does not use the value, but returns it in the response to the command.

NAME: The name the STARTINTERVIEW command uses to reference the study that you are loading. It is alphanumeric and required.

URL: The address of an XML document that defines a study. The document must validate successfully against the study.dtd file (for details, see Appendix A) and must be a valid Universal Resource Locator (URL) value. The machine where Precision Choice is running must have access to this URL and privileges to read the document this address indicates. The XML document that this URL points to must conform to the study definition.

Response Format 1 (Successful)

<CMDRESPONSE CID="1" STATUS="OK"/>

Elements of Response Format 1

Note: XML is case-sensitive.

CMDRESPONSE: This tag contains the Precision Choice response to a command.

CID: The command identification defined in the command that generated this response. This value can be used to associate the response with the command that caused the response.

STATUS: If the command was successful, the value is "OK".

Response Format 2 (Error)

<CMDRESPONSE CID="878" STATUS="ERR">

<MESSAGE ID="234">Duplicate study.</MESSAGE>

</CMDRESPONSE>

Elements of Response Format 2

Note: XML is case-sensitive.

CMDRESPONSE: Tag containing the error message.

CID: The command identification defined in the command that generated the response. This value can be used to associate the response with the command that caused the response.

STATUS: If the command was unsuccessful, the value is "ERR".

MESSAGE: Tag containing the error message.

ID: The error's ID number. This ID can be used to perform an action or to convert the error message into text customized for your application.

SETCATALOG Command

Use the SETCATALOG command to load a catalog into Precision Choice. Interviews are based upon studies and catalogs that must be loaded into Precision Choice before an interview can reference them. Although defined catalogs are normally loaded when Precision Choice is started, a catalog can be loaded as part of an interview by preceding the STARTINTERVIEW command with the SETCATALOG command.

The SETCATALOG command has two formats. One format allows a catalog file to be loaded into Precision Choice. This format is a tag with only attributes. The other format allows the catalog file to be defined and loaded directly in the SETCATALOG command. This format defines the SETCATALOG command with two attributes and one element that contains the catalog definition.

Format 1 of the SETCATALOG Command

```
<SETCATALOG CID="1 NAME="computerproducts">
<catalog name="computerproducts "version="1.0">
<product
id="1"
name="Compaq Presario 1900">
<attribute id="Brand">Compaq</attribute>
<attribute id="Price">2950</attribute>
<attribute id="RAM">128</attribute>
<attribute id="Speed">366</attribute>
<attribute id ="Processor">Pentium II</attribute>
<references>
<reference type="image">someimage.jpg</reference>
<reference
type="details">inspiron7000.html</reference>
<references>
</product>
```

</catalog>

</SETCATALOG>

Elements of Format 1 of the SETCATALOG Command

Note: XML is case-sensitive.

SETCATALOG: Tag containing the catalog definition.

CID: The command identification returned with the response to this command. This value is used to associate the Precision Choice response with the command that caused it. It is alphanumeric and required. Precision Choice does not use the value but returns it in the response to the command.

NAME: The name the STARTINTERVIEW command uses to reference the catalog you are loading. It is alphanumeric and required.

The catalog Definition

The catalog definition defines the products available for an interview. At the end of an interview, Precision Choice recommends products that meet an end-user's criteria. This command defines available products to search through. There must be at least one value for every attribute defined in the study for each product in the catalog. The catalog can define attributes that are not in the study. All attributes of a product are returned in the product recommendations. The catalog definition is never sent directly to Precision Choice. The reference tag is provided to store other types of data with the product. The references are returned with the GETRECOMMENDATIONS command, and the application can use them.

Format of the catalog Definition

<catalog name="computerproducts" version="1.0">

<product

id="1"

name="Compaq Presario 1900">

<attribute id="Brand">Compaq</attribute>

<attribute id="Price">2950 </attribute>

<attribute id="RAM">128 </attribute>

<attribute id="Speed">366 </attribute>

<attribute id = "Processor">Pentium II</attribute>

<references>
<reference
type="image">someimage.jpg</reference>
<reference
type="details">inspiron7000.html</reference>
</references>
</product>
<product
id="2"
name="Compaq Prosignia 150">
<attribute id="Brand">Compaq </attribute>
<attribute id="Price">1800 </attribute>
<attribute id="RAM">32 </attribute>
<attribute id="Speed">350 </attribute>
<attribute id = "Processor">K-6</attribute>
</product>
</catalog>

Elements of the catalog Definition

Note: XML is case-sensitive.

catalog: Contains all products for a catalog.

name (for the catalog tag): Catalog's name. This name is for your purposes only. Precision Choice does not use it. This is not the same as the NAME element in the SETCATALOG command.

version: Defines the catalog's version. It is for your purposes only. Precision Choice does not use it.

product: Tag that defines a product. All attributes and references for a product are defined in this element, as well as a text name of the product. This value is intended to be displayed to the end-user.

id (for the product tag): The product key. The value must be unique in the catalog. This element is required and alphanumeric. Precision Choice uses this value as a unique identifier for each product in the catalog.

name (for the product tag): Product name. This element is required and alphanumeric. It can be displayed to the user.

attribute: Defines an attribute of the product. An attribute is a product feature that end-users may find important. A product must have a product attribute that corresponds with each attribute defined in the study, though it may have additional attributes not defined in the study. The value of the product attribute is defined in this element.

id (for the attribute tag): If this attribute provides information for an attribute defined in the study, the value should match the desc of the attribute defined in a study. It is required.

references: Tag that contains all references for a product. It is not required.

reference: Tag that defines a product reference: it could be an HTML page, jpeg file, or anything associated with the product. This tag is optional and can only be contained in a references tag. Precision Choice does not use this information, but simply returns it in the response to the GETRECOMMENDATIONS command.

type: Defines the type of reference. Precision Choice does not use this, but it is available for an application to base processing on. This value is returned in the response to the GETRECOMMENDATIONS command.

Format 2 of the SETCATALOG Command

```
<SETCATALOG
CID="2"
NAME="computerproducts"
URL="http://www.onlineinsight.com/products.xml" />
```

Elements of Format 2 of the SETCATALOG Command

Note: XML is case-sensitive.

SETCATALOG: Command tag for setting the catalog. In this case, it does not contain additional attributes.

CID: The command identification returned with the response to this command. This value can be used to associate the Precision Choice response with the command that caused it. It is alphanumeric and required. Precision Choice does not use this value but returns it in the response to the command.

NAME: The name the STARTINTERVIEW command uses to reference the catalog you are loading. It is alphanumeric and required.

URL: The address of an XML document that defines a study. The document must validate successfully against the catalog.dtd file (for details, see Appendix A) and must be a valid Universal Resource Locator (URL) value. The machine where Precision Choice is running must have access to this URL and privileges to read the document this address indicates. The XML document that this URL points to must conform to the catalog definition.

Response Format 1(Successful)

<CMDRESPONSE CID="1" STATUS="OK"/>

Elements of Response Format 1

Note: XML is case-sensitive.

CMDRESPONSE: This tag contains the Precision Choice response to a command.

CID: The command identification defined in the command that generated the response. This value can be used to associate the response with the command that caused the response.

STATUS: If the command was successful, the value is "OK".

Response Format 2 (Error)

<CMDRESPONSE CID="878" STATUS="ERR">

<MESSAGE ID="234">Duplicate catalog.</MESSAGE>

</CMDRESPONSE>

Elements of Response Format 2

Note: XML is case-sensitive.

CMDRESPONSE: Tag containing the error message.

CID: The command identification defined in the command that generated the response. This value can be used to associate the response with the command that caused the response.

STATUS: If the command was unsuccessful, the value is "ERR".

MESSAGE: This tag contains the error message.

ID: The error's ID number. This ID can be used to perform an action or to convert the error message into text customized for your application.

GETSTATUS Command

Use the GETSTATUS command to obtain statistics on a cluster of Precision Choice engines. You can see how many active interviews are in progress and how much memory is free in the cluster.

Format of the GETSTATUS Command

<GETSTATUS CID="5"/>

Elements of the GETSTATUS Command

Note: XML is case-sensitive.

GETSTATUS: Tag containing the command that obtains statistics on a cluster of Precision Choice engines.

CID: The command identification returned with the response to this command. You can use this value to associate the Precision Choice response with the command that caused it. It is alphanumeric and required. Precision Choice does not use the value but returns it in the response to the command.

Response Format 1(Successful)

<CMDRESPONSE CID="5" STATUS="OK" >

<NUMINTERVIEWS>27</NUMINTERVIEWS>

<FREEMEMORY>1000000</FREEMEMORY>

</CMDRESPONSE>

Elements of Response Format 1

Note: XML is case-sensitive.

CMDRESPONSE: Tag containing statistics for a cluster of Precision Choice engines.

CID: The command identification defined in the command that generated the response. This value can be used to associate the response with the command that caused the response.

STATUS: If the command was successful, the value is "OK".

NUMINTERVIEWS: Tag returning the number of active interviews for a particular cluster of Precision Choice engines.

FREEMEMORY: Tag returning the amount of free memory available in the cluster of Precision Choice engines.

Response Format 2 (Error)

<CMDRESPONSE CID="5" STATUS="ERR">

<MESSAGE ID="100">Precision Choice not responding.

</MESSAGE>

</CMDRESPONSE>

Elements of Response Format 2

CMDRESPONSE: Tag containing the error message.

CID: The command identification defined in the command that generated the response. This value can be used to associate the response with the command that caused the response.

STATUS: If the command was unsuccessful, the value is "ERR".

MESSAGE: Tag containing the error message.

ID: The error's ID number. This ID can be used to perform an action or to convert the error message into text customized for your application.

Interview Commands

Use interview commands to conduct an interview with Precision Choice. All the interview commands are grouped inside a batch tag. Herein, the commands are listed separately, but they can be combined inside a BATCH tag and sent all at once.

STARTINTERVIEW

GETNEXTQUESTIONSET

SETRESPONSES

GETSCORES

GETRECOMMENDATIONS

ENDINTERVIEW

The BATCH Tag

All Interview commands must be located inside a BATCH element. There may be one or more than one Interview commands included in each BATCH element. There are three elements in the BATCH tag.

Elements of the BATCH Tag

BATCH: Tag containing a batch of commands. It must be provided for any batch command.

UID: User identification the application provides. Precision Choice associates interview results with this User ID. It is alphanumeric and required.

INTERVIEWID: This element associates commands with a particular interview. It is not needed for the batch with a STARTINTERVIEW command but is required for all subsequent batches. If the INTERVIEWID is not defined in the batch tag, Precision Choice assumes that the batch is a new interview. Precision Choice generates a unique ID and assigns it to the INTERVIEWID. It is returned in the batch response. All future batches for the interview must contain the returned INTERVIEWID, or the batch is rejected.

STARTINTERVIEW Command

Use the STARTINTERVIEW command to begin an interview: it must precede any other Interview commands. You tell Precision Choice which previously loaded study and catalog to use in this interview.

User-specific attributes can also be defined for this interview. User-specific attributes can be dynamically loaded at interview time and can apply only to the specific interview. For example, if there is an attribute of a product called "Cost of Ownership" that differs for types of end-users, you can use a user-specific attribute to model this.

The attribute and value for each product for a specific type of user can be loaded during the STARTINTERVIEW command. Precision Choice calculates the levels of a USERATTR.

In this version of Precision Choice, the values of a USERATTR must be able to be interpolated and numeric. Precision Choice calculates four levels for the USERATTR: the lowest value, the highest value, and two equidistant values of the product values provided in this tag. There must be a value for every product in the catalog. The RANKORDER tag determines the order of the levels.

Format of the STARTINTERVIEW Command

<BATCH UID="678" >

<STARTINTERVIEW

```
CID="3"
STUDY="noteboookstudy"
CATALOG="notebookproducts">
<USERATTR
NAME="Cost of Ownership"
UNIT="currency"
INTERPOLATE="yes"
APRIORI="yes"
EXCLUDEFROMPAIRS="no"
RANKORDER="ascending">
<PRODKEY ID="5" VALUE="2500.00"/>
<PRODKEY ID="8" VALUE="1000.00"/>
</USERATTR>
</STARTINTERVIEW>
</BATCH>
```

Elements of the STARTINTERVIEW Command

Note: XML is case-sensitive.

STARTINTERVIEW: Tag specifying the study and catalog to use in this interview. This tag may contain any user-specific attributes.

CID: The command identification returned with the response to this command. You can use this value to associate the Precision Choice response with the command that caused it. It is alphanumeric and required. Precision Choice does not use the value, but returns it in the response to the command.

STUDY: Name of the study to use for this interview. This name must match the name in the SETSTUDY command used to load the study. It is required.

CATALOG: Name of the catalog to use for this interview. This name must match the name in the SETCATALOG command used to load the catalog. It is required.

USERATTR: Tag defining an attribute for this interview, used only for this interview. This allows your application dynamically to develop an attribute and its levels for a particular user. These attributes are not visible to any other interview.

NAME: Text description of the attribute. This information becomes part of the questions Precision Choice generates. It is required for each attribute. It is alphanumeric and can contain multiple words.

UNIT: Defines how to display the attribute's levels. The unit must be defined in the units.xml file and is required.

INTERPOLATE: Value telling Precision Choice whether values of the levels can be interpolated. Typically, attributes such as "Brand" are not interpolated; attributes such as "Price" are interpolated. It is not required. It can be either "Yes" or "No." If it is not present, Precision Choice assumes that the value is "No."

APRIORI: If the levels of an attribute have an assumed choice, this value should be "Yes." An example is "Price": if provided a choice, virtually everyone wants the lowest price. It is not required, and the default value is "No." If this tag's value is "Yes," RANKORDER must be specified in the USERATTR tag.

EXCLUDEFROMPAIRS: Tag telling Precision Choice whether to exclude this attribute in the pairs questions. Value can be either "Yes" or "No." It is not required, and the default value is "No."

RANKORDER: Value is used in conjunction with the APRIORI switch. If APRIORI is "Yes," RANKORDER is required. RANKORDER tells Precision Choice how to order the levels of the user attribute: values are "ascending" and "descending." If the value is "ascending," Precision Choice orders the levels in ascending order, and the best value is the lowest. If the value is "descending," Precision Choice orders the levels in descending order, and the best value is the highest.

PRODKEY: Each product in the catalog must have a unique value defined for this attribute. You define a value for each product with this tag.

ID: The key of the product. It must be unique in the catalog. This element is required and alphanumeric. It must match the ID defined in the catalog definition.

VALUE: The value for the attribute of this product. It is required.

Response Format 1 (Successful)

```
<BATCHRESPONSE UID="678" INTERVIEWID="123">
<CMDRESPONSE CID="3" STATUS="OK"/>
</BATCHRESPONSE>
```

Elements of Response Format 1

Note: XML is case-sensitive.

BATCHRESPONSE: The response to a batch command. A BATCHRESPONSE contains a CMDRESPONSE for each command in the batch.

UID: Value matching the UID submitted in the previous corresponding batch tag.

INTERVIEWID: Value matching the INTERVIEWID submitted in the previous corresponding batch tag. If no INTERVIEWID is specified on the batch tag, Precision Choice generates a unique ID and sends it back through this element.

CMDRESPONSE: This tag contains the Precision Choice response to a command.

CID: The command identification defined in the command that generated the response. You can use this value to associate the response with the command that caused the response.

STATUS: If the command was successful, the value is "OK".

Response Format 2 (Error)

<BATCHRESPONSE UID="678" INTERVIEWID="123">
<CMDRESPONSE CID="878" STATUS="ERR">
<MESSAGE ID="412">STUDY not found.</MESSAGE>
</CMDRESPONSE>
<BATCHRESPONSE>

Elements of Response Format 2

Note: XML is case-sensitive.

BATCHRESPONSE: Response to a batch command. A BATCHRESPONSE contains a CMDRESPONSE for each command in the batch.

UID: Value matching the UID submitted in the previous corresponding batch tag.

INTERVIEWID: Value matching the INTERVIEWID submitted in the previous corresponding batch tag. If no INTERVIEWID is specified on the batch tag, Precision Choice generates a unique ID and sends it back through this element.

CMDRESPONSE: Tag containing the error message.

CID: The command identification defined in the command that generated the response. You can use this value to associate the response with the command that caused the response.

STATUS: If the command was unsuccessful, the value is "ERR".

MESSAGE: Tag containing the error message.

ID: The error's ID number. This ID can be used to perform an action or to convert the error message into text customized for your application.

GETNEXTQUESTIONSET Command

Use the GETNEXTQUESTIONSET command to get questions from Precision Choice. Precision Choice can generate four types of questions:

ratings

importance

pairs

calibration

Questions are generated in the order listed above. A question must be generated before a SETREPONSE for the question can be sent to Precision Choice. All ratings and Importance questions must be generated and answered before the GETSCORES command can be sent to Precision Choice.

<BATCH UID="678" INTERVIEWID="123">

<GETNEXTQUESTIONSET CID="3" MAX="5"/>

</BATCH>

Elements of the GETNEXTQUESTIONSET

Command

Note: XML is case-sensitive.

GETNEXTQUESTIONSET: Tag that causes Precision Choice to generate and return interview questions. If the interview is complete, an empty question set is returned.

CID: The command identification returned with the response to this command. This value can be used to associate the Precision Choice response with the command that caused it. It is alphanumeric and required. Precision Choice does not use the value but returns it in the response to the command.

MAX: Specifies the maximum number of questions to return with this attribute. Precision Choice generates questions and returns up to the maximum number you specify with this element. Precision Choice can generate up to a certain amount of questions for each section. For ratings questions, the maximum number of questions that Precision Choice can generate equals the number of attributes in the study minus the number of a priori attributes in the study.

Precision Choice can generate ratings questions for all non-a priori attributes defined in the study. When an end-user has answered all ratings questions, Precision Choice can generate importance questions for all attributes defined in the study. Pairs questions are created individually; each pairs question is based upon an end-user's answers to previous questions. Once all pairs questions have been generated and answered, Precision Choice can create calibration questions. The number of calibration questions that Precision Choice can generate is defined in PrecisionChoice.properties.

By default, Precision Choice generates pairs questions in the following way. Precision Choice asks seven total pairs questions. The first three pairs questions compare sample products with two attributes each. The next three questions compare sample products with three attributes each. The last question compares two products with four attributes each. By default, Precision Choice does not ask calibration questions.

Response Format 1(Ratings question)

```
<BATCHRESPONSE UID="678" INTERVIEWID="123">
<CMDRESPONSE CID="3" STATUS="OK">
<QUESTIONSET>
<RATINGQUESITON ID="12" >
<ATTRIBUTE ID="134" NAME="CPU" >
<LEVEL ID= "845" VALUE= "P II 233" />
<LEVEL ID= "846" VALUE= "P III 360" />
<LEVEL ID= "847" VALUE= "P III 450" />
<LEVEL ID= "848" VALUE= "PPC 360" />
<LEVEL ID= "849" VALUE= "Sparc Ultra 350" />
</ATTRIBUTE>
```

<DEFAULTQUESTION>Rate these attributes.

</DEFAULTQUESTION>

</RATINGQUESITON>

</QUESTIONSET>

</CMDRESPONSE>

</BATCHRESPONSE>

Elements of Response Format 1

Note: XML is case-sensitive.

BATCHRESPONSE: The response to a batch command. A BATCHRESPONSE contains a CMDRESPONSE for each command in the batch.

UID: Value matching the UID submitted in the previous corresponding batch tag.

INTERVIEWID: Value matching the INTERVIEWID submitted in the previous corresponding batch tag. If no INTERVIEWID is specified on the batch tag, Precision Choice generates a unique ID and sends it back through this element.

CMDRESPONSE: Tag containing the questions Precision Choice generates.

CID: The command identification defined in the command that generated the response. You can use this value to associate the response with the command that caused the response.

STATUS: If the command was successful, the value is "OK".

QUESTIONSET: Tag containing the questions.

RATINGSQUESTION: Tag identifying a Ratings question, i.e., a question that asks a user to identify how much he or she prefers the different levels of an attribute.

ID (for RATINGSQUESTION tag): The question's ID. In the SETRESPONSE command, this identifies the question being answered. This ID can be used to answer this question any time during the interview, and the end-user can answer the question more than once. For example, Precision Choice may generate question 1 and then receive an answer, and then do the same for questions 2 and 3. After this, Precision Choice generates question 4.

Imagine that before the end-user sends the answer to this question, he or she moves back to question 2 and re-answers it. Precision Choice over-writes the previous

answer for question 2 and then removes the generated questions 3 and 4. The next GETRECOMMENDATIONSET command causes Precision Choice to generate a new question 3, and the interview proceeds from that point.

ATTRIBUTE: Tag containing the levels to display to your user when Precision Choice asks a ratings question.

ID (for ATTRIBUTE tag): The attribute's ID that Precision Choice returns. You should use this ID in the SETRESPONSE command to reference this attribute.

NAME: Description of the attribute defined in the study and displayed to the end-user.

LEVEL: Tags returning the attribute's levels defined in the study.

ID (for LEVEL tag): The level's ID that Precision Choice returns. It should be used in the SETRESPONSE command to reference this level.

VALUE: The level's value defined in the study. It should be displayed to the user when Precision Choice presents this question.

DEFAULTQUESTION: The default question defined in the study that may be displayed to the end-user.

Response Format 2 (Importance question)

```
<BATCHRESPONSE UID="678" INTERVIEWID="123">
<CMDRESPONSE CID="3" STATUS="OK">
<QUESTIONSET>
<IMPORTANCEQUESTION ID="13" >
<ATTRIBUTE
ID= "134"
NAME="Processor"
BEST="Sparc Ultra 350"
WORST="P II 233" />
<ATTRIBUTE
ID= "135"
NAME="Hard drive"
BEST="50"
WORST="3" />
```

<DEFAULTQUESTION>
Rank the importance of these attributes.
</DEFAULTQUESTION>
</IMPORTANCEQUESTION>
</QUESTIONSET>
</CMDRESPONSE>
</BATCHRESPONSE>

Elements of Response Format 2

Note: XML is case-sensitive.

BATCHRESPONSE: Response to a batch command. A BATCHRESPONSE contains a CMDRESPONSE for each command in the batch.

UID: Value matching the UID submitted in the previous corresponding batch tag.

INTERVIEWID: Value matching the INTERVIEWID submitted in the previous corresponding batch tag. If no INTERVIEWID is specified on the batch tag, Precision Choice generates a unique ID and sends it back through this element.

CMDRESPONSE: Tag containing the questions Precision Choice generates.

CID: The command identification defined in the command that generates the response. This value can be used to associate the response with the command that caused the response.

STATUS: If the command was successful, the value is "OK".

QUESTIONSET: Tag containing the questions.

IMPORTANCEQUESTION: Tag identifying an importance question, i.e., a question that asks an end-user to quantify how important a particular attribute is in his or her decision-making process.

ID (for IMPORTANCEQUESTION tag): The question's ID. In the SETRESPONSE command, this identifies the question being answered. This ID can be used to answer this question any time during the interview, and the end-user can answer the question more than once. For example, Precision Choice may generate question 1 and then receive an answer, and then do the same for questions 2 and 3. After this, Precision Choice generates question 4.

Imagine that before the end-user sends the answer to this question, he or she moves back to question 2 and re-answers it. Precision Choice over-writes the previous answer for question 2 and then removes the generated questions 3 and 4. The next GETRECOMMENDATIONSET command causes Precision Choice to generate a new question 3, and the interview proceeds from that point.

ATTRIBUTE: Tag defining an attribute used in an Importance question.

ID (for ATTRIBUTE tag): The attribute's ID that Precision Choice returns. You should use it in the SETRESPONSE command to reference this attribute.

NAME: Description of the attribute defined in the study and displayed to the end-user.

BEST: The best level of the attribute. This value can be displayed to the end-user to help him or her determine how important this attribute is in the decision-making process.

WORST: The worst level of the attribute. This value can be displayed to the end-user to help him or her determine how important this attribute is in the decision-making process.

DEFAULTQUESTION: The default question defined in the study that may be displayed to the end-user.

Response Format 3 (Pairs question)

```
<BATCHRESPONSE UID="678" INTERVIEWID="123">
<CMDRESPONSE CID="3" STATUS="OK">
<QUESTIONSET>
<PAIRSQUESTION ID="14">
<SAMPLE SIDE="low">
<ATTRIBUTE ID="134" NAME="Processor" >
<LEVEL VALUE="P II 300"/>
</ATTRIBUTE>
<ATTRIBUTE ID="135" NAME="Hard drive" >
<LEVEL VALUE="10.2"/>
</ATTRIBUTE>
</SAMPLE>
```

<SAMPLE SIDE="high">
<ATTRIBUTE ID="134" NAME="Processor">
<LEVEL VALUE="P II 450"/>
</ATTRIBUTE>
<ATTRIBUTE ID="135" NAME="Hard drive" >
<LEVEL VALUE="5.2"/>
</ATTRIBUTE>
</SAMPLE>
<DEFAULTQUESTION>Which pair is better?
</DEFAULTQUESTION>
</PAIRSQUESTION>
</QUESTIONSET>
</CMDRESPONSE>
</BATCHRESPONSE>

Elements of Response Format 3

Note: XML is case-sensitive.

BATCHRESPONSE: Response to a batch command. A BATCHRESPONSE contains a CMDRESPONSE for each command in the batch.

UID: Value matching the UID submitted in the previous corresponding batch tag.

INTERVIEWID: Value matching the INTERVIEWID submitted in the previous corresponding batch tag. If no INTERVIEWID is specified on the batch tag, Precision Choice generates a unique ID and sends it back through this element.

CMDRESPONSE: Tag containing the questions Precision Choice generates.

CID: The command identification defined in the command that generates the response. You can use this value to associate the response with the command that causes the response.

STATUS: If the command was successful, the value is "OK".

QUESTIONSET: Tag containing the questions.

PAIRSQUESTION: Tag identifying a pairs question, i.e., a question that shows two sample products and asks an end-user to rate his or her user preference between the

two sample products on a scale of 1 to 9. The samples are identified by "low" and "high." The greater the user prefers the "low" sample, the closer to 1 his or her response should be. The greater the user prefers the "high" sample, the closer to 9 his or her response should be. If the user has no preference between the two products, the answer should be five. This is required and must be a whole number.

ID (for PAIRSQUESTION tag): The question's ID. In the SETRESPONSE command, this identifies the question being answered. This ID can be used to answer this question any time during the interview, and the end-user can answer the question more than once. For example, Precision Choice may generate question 1 and then receive an answer, and then do the same for questions 2 and 3. After this, Precision Choice generates question 4.

Imagine that before the end-user sends the answer to this question, he or she moves back to question 2 and re-answers it. Precision Choice over-writes the previous answer for question 2 and then removes the generated questions 3 and 4. The next GETRECOMMENDATIONSET command causes Precision Choice to generate a new question 3, and the interview proceeds from that point.

SAMPLE: Tag identifying the sample product.

SIDE: Values can be "low" or "high."

ATTRIBUTE: Tag defining an attribute of the sample product.

ID (for ATTRIBUTE tag): The attribute's ID.

NAME: Description of the attribute defined in the study and displayed to the user.

LEVEL: Tag defining the attribute's level in this sample product.

VALUE: Value of the level displayed to the user.

DEFAULTQUESTION: Default question defined in the study that may be displayed to the user.

Response Format 4 (Calibration question)

<BATCHRESPONSE UID="678" INTERVIEWID="123">

<CMDRESPONSE CID="3" STATUS="OK">

<QUESTIONSET>

<CALIBRATIONQUESTION ID="12">

<SAMPLE>
<ATTRIBUTE
ID="12"
NAME="BRAND"
VALUE="Sony" />
<ATTRIBUTE
ID="13"
NAME="Hard Drive"
VALUE="10" />
</SAMPLE>
<DEFAULTQUESTION>Would you buy this?
</DEFAULTQUESTION>
</CALIBRATIONQUESTION>
</QUESTIONSET>
</CMDRESPONSE>
</BATCHRESPONSE>

Elements of Response Format 4

Note: XML is case-sensitive.

BATCHRESPONSE: The response to a batch command. A BATCHRESPONSE contains a CMDRESPONSE for each command in the batch.

UID: Value matching the UID submitted in the previous corresponding batch tag.

INTERVIEWID: Value matching the INTERVIEWID submitted in the previous corresponding batch tag. If no INTERVIEWID is specified on the batch tag, Precision Choice generates a unique ID and sends it back through this element.

CMDRESPONSE: Tag containing the questions Precision Choice generates.

CID: The command identification defined in the command that generates the response. You can use this value to associate the response with the command that caused the response.

STATUS: If the command was successful, the value is "OK".

QUESTIONSET: Tag containing the questions.

CALIBRATIONQUESTION: Tag identifying a calibration question, i.e., a question used to determine how valid an end-user's responses are. Precision Choice presents sample products, and asks the end-user to specify how likely it is that he or she would purchase the product on a scale of 0 to 100.

ID (for CALIBRATIONQUESTION tag): The question's ID. In the SETRESPONSE command, this identifies the question being answered. This ID can be used to answer this question any time during the interview, and the end-user can answer the question more than once. For example, Precision Choice may generate question 1 and then receive an answer, and then do the same for questions 2 and 3. After this, Precision Choice generates question 4.

Imagine that before the end-user sends the answer to this question, he or she moves back to question 2 and re-answers it. Precision Choice over-writes the previous answer for question 2 and then removes the generated questions 3 and 4. The next GETRECOMMENDATIONSET command causes Precision Choice to generate a new question 3, and the interview proceeds from that point.

SAMPLE: Tag identifying the sample product.

ATTRIBUTE: Tag defining an attribute of the sample product.

ID (for ATTRIBUTE tag): The attribute's ID.

NAME: Description of the attribute defined in the study and displayed to the end-user.

VALUE: Value of the attribute displayed to the user.

DEFAULTQUESTION: Default question defined in the study that may be displayed to the user.

Response Format 5 (Empty question)

```
<BATCHRESPONSE UID="678" INTERVIEWID="123">
<CMDRESPONSE CID="3" STATUS="OK">
<QUESTIONSET/>
</CMDRESPONSE>
</BATCHRESPONSE>
```

Elements of Response Format 5

Note: XML is case-sensitive.

BATCHRESPONSE: The response to a batch command. A BATCHRESPONSE contains a CMDRESPONSE for each command in the batch.

UID: Value matching the UID submitted in the previous corresponding batch tag.

INTERVIEWID: Value matching the INTERVIEWID submitted in the previous corresponding batch tag. If no INTERVIEWID is specified on the batch tag, Precision Choice generates a unique ID and sends it back through this element.

CMDRESPONSE: Tag containing the questions Precision Choices generates.

CID: The command identification defined in the command that generates the response. You can use this value to associate the response with the command that caused the response.

STATUS: If the command was successful, the value is "OK".

QUESTIONSET: Tag indicating that there are no more questions to ask in the interview.

Response Format 6 (Error)

<BATCHRESPONSE UID="678" INTERVIEWID="123">
<CMDRESPONSE CID="878" STATUS="ERR">
<MESSAGE ID="412">Study not found.</MESSAGE>
</CMDRESPONSE>
<BATCHRESPONSE>

Elements of Response Format 6

Note: XML is case-sensitive.

BATCHRESPONSE: The response to a batch command. A BATCHRESPONSE contains a CMDRESPONSE for each command in the batch.

UID: Value matching the UID submitted in the previous corresponding batch tag.

INTERVIEWID: Value matching the INTERVIEWID submitted in the previous corresponding batch tag. If no INTERVIEWID is specified on the batch tag, Precision Choice generates a unique ID and sends it back through this element.

CMDRESPONSE: Tag containing the questions Precision Choice generates.

CID: The command identification defined in the command that generates the response. You can use this value to associate the response with the command that caused the response.

STATUS: If the command was unsuccessful, the value is "ERR".

MESSAGE: Tag containing the error message.

ID: The error's ID number. This ID can be used to perform an action or to convert the error message into text customized for your application

SETRESPONSES Command

Use the SETRESPONSES command to provide answers to questions that Precision Choice generates.There are four types of questions in Precision Choice: ratings, importance, pairs, and calibration. There are four corresponding responses: ratings response, importance response, pairs response, and calibration response. An ID for each question is returned in the response to GETNEXTQUESTIONSET. The value of that ID must be part of the SETRESPONSES command.

The ID tells Precision Choice which question that you are providing an answer for. You can also use this ID to change the answer to the question. Precision Choice restarts the interview from that point.

Precision Choice uses the answers to questions to understand the end-user's preferences about the product. You can request scores and product recommendations any time in the interview after the rating and importance questions have been generated and answered. However, the more questions the end-user answers, the better the product recommendations are. The SETRESPONSES command must be inside a BATCH. The INTERVIEWID must match the INTERVIEWID returned in the response to the STARTINTERVIEW command. Multiple SETRESPONSES can be included in one batch if all the responses are for one interview. This is useful for collecting a group of questions from multiple GETNEXTQUESTIONSETS commands, presenting all the questions at once to the end-user, and sending all of the end-user's answers at one time to Precision Choice.

Format 1 (Answer a Ratings Question)

<BATCH UID="678" INTERVIEWID="123">

<SETRESPONSES CID="15">

<RATINGRESPONSE ID= "12" >
<LEVEL ID= "845" RATING= "2" />
<LEVEL ID= "846" RATING= "2" />
<LEVEL ID= "847" RATING= "2" />
<LEVEL ID= "848" RATING= "2" />
<LEVEL ID= "849" RATING= "2" />
</RATINGRESPONSE>
</SETRESPONSES>
</BATCH>

Elements of Format 1

Note: XML is case-sensitive.

SETRESPONSES: Element containing an answer to a question.

CID: The command identification returned with the response to this command. You can use this value to associate the Precision Choice response with the command that caused it. It is alphanumeric and required. Precision Choice does not use this value but simply returns it in the response to the command.

RATINGRESPONSE: Identifies the question you are answering and contains the answer for the question.

ID (for the RATINGRESPONSE tag): The question's ID. You should use this value in the SETRESPONSE command to identify the question being answered. This value is provided in the response to the GETNEXTQUESTIONSET command. It is required.

LEVEL: Empty tags identifying the levels of the attribute and the value the end-user assigned to each. They are defined in the study and provided in the response to the GETNEXTQUESTIONSET command. You should have one LEVEL element for each level returned from the GETNEXTQUESTIONSET command.

ID (for the LEVEL tag): The Level's ID. This identifies the level to Precision Choice and is provided in the response to the GETNEXTQUESTIONSET command.

RATING: Rating the user assigned to this level. The value should be a whole number between 1 and 5, inclusive: 1 is the worst rating, and 5 is the best rating. This value is required.

Format 2: Answer an Importance Question

<BATCH UID="678" INTERVIEWID="123">

<SETRESPONSES CID="15">

<IMPORTANCERESPONSE ID= "12" >

<ATTRIBUTE ID= "134" NAME="Brand" RATING= "2" />

</IMPORTANCERESPONSE>

</SETRESPONSES>

</BATCH>

Elements of Format 2

Note: XML is case-sensitive.

SETRESPONSES: Contains an answer to a question.

CID: The command identification returned with the response to this command. You can use this value to associate the Precision Choice response with the command that caused it. It is alphanumeric and required. Precision Choice does not use the value, but returns it in the response to the command.

IMPORTANCERESPONSE: Command identifying which question is being answered and containing the answer for the question.

ID (for the IMPORTANCERESPONSE tag): The question's ID. You should use this value in the SETRESPONSE command to tell Precision Choice which question you are answering. This value is provided in the response to the GETNEXTQUESTIONSET command. It is required.

ATTRIBUTE: Tag identifying the attribute and the value the end-user assigned to it. It is required.

ID (for the ATTRIBUTE tag): The attribute's ID. This ID identifies the level to Precision Choice. This element is provided in the response to the GETNEXTQUESTIONSET command. It is required.

NAME: The attribute's name provided in the response to the GETNEXTQUESTIONSET command. It is required.

RATING: Rating that the user assigns to this level. The value should be a whole number between 1 and 4, inclusive: 1 represents the least importance and 4 represents the most importance. This value is required.

Format 3 (Answer a Pairs Question)

<BATCH UID="678" INTERVIEWID="123">

<SETRESPONSES CID="15">

<PAIRSRESPONSE ID= "14" RATING = "2"/>

</SETRESPONSES>

</BATCH>

Elements of Format 3

Note: XML is case-sensitive.

SETRESPONSES: Contains an answer to a question.

CID: The command identification returned with the response to this command. You can use this value to associate the Precision Choice response with the command that caused it. It is alphanumeric and required. Precision Choice does not use the value but returns it in the response to the command.

PARISRESPONSE: Tag identifying which question is being answered and containing the answer for the question.

ID: The question's ID. You should use this value in the SETRESPONSE command to tell Precision Choice which question you are answering. This value is provided in the response to the GETNEXTQUESTIONSET command. It is required.

RATING: The rating the end-user assigns to this level. The value should be a whole number between 1 and 9, inclusive. The pairs question asks the end-user to chose between two sample products. The more the end-user prefers the "low" product, the closer the rating should be to 1. The more the end-user prefers the "high" product, the closer the rating should be to 9. If the end-user has no preference between the two products, the rating should be 5.

Format 4 (Answer a Calibration Question)

<BATCH UID="678" INTERVIEWID="123">

<SETRESPONSES CID="15">

<CALIBRATIONRESPONSE ID= "14" RATING = "50"/>

</SETRESPONSES>

</BATCH>

Elements of Format 4

Note: XML is case-sensitive.

SETRESPONSES: Contains an answer to a question.

CID: The command identification returned with the response to this command. You can use this value to associate the Precision Choice response with the command that caused it. It is alphanumeric and required. Precision Choice does not use the value but simply returns it in the response to the command.

CALIBRATIONRESPONSE: Tag identifying which question you are answering and containing the answer for the question.

ID: The question's ID. You should use this value in the SETRESPONSE command to identify the question you are answering. This value is provided in the response to the GETNEXTQUESTIONSET command and is required.

RATING: Rating the end-user assigned to this level. The value should be a whole number between 0 and 100, inclusive. The calibration question presents a sample product with attributes based upon the end-user's interview and is used to determine how valid the end-user's answers are in the interview.

Response Format 1 (Successful)

<BATCHRESPONSE UID="678" INTERVIEWID="123">

<CMDRESPONSE CID="3" STATUS="OK"/>

</BATCHRESPONSE>

Elements of Response Format 1

Note: XML is case-sensitive.

BATCHRESPONSE: Response to a batch command. A BATCHRESPONSE contains a CMDRESPONSE for each command in the batch.

UID: Value matching the UID submitted in the previous corresponding batch tag.

INTERVIEWID: Value matching the INTERVIEWID submitted in the previous corresponding batch tag. If no INTERVIEWID is specified on the batch tag, Precision Choice generates a unique ID and sends it back through this element.

CMDRESPONSE: This tag contains the Precision Choice response to a command.

CID: The command identification defined in the command that generated the response. You can use this value to associate the response with the command that caused the response.

STATUS: If the command was successful, the value is "OK".

Response Format 2 (Error)

<BATCHRESPONSE UID="678" INTERVIEWID="123">

<CMDRESPONSE CID="878" STATUS="ERR">

<MESSAGE ID="415">Invalid answer.</MESSAGE>

</CMDRESPONSE>

<BATCHRESPONSE>

Elements of Response Format 2

Note: XML is case-sensitive.

BATCHRESPONSE: Response to a batch command. A BATCHRESPONSE contains a CMDRESPONSE for each command in the batch.

UID: Value matching the UID submitted in the previous corresponding batch tag.

INTERVIEWID: Value matching the INTERVIEWID submitted in the previous corresponding batch tag. If no INTERVIEWID is specified on the batch tag, Precision Choice generates a unique ID and sends it back through this element.

CMDRESPONSE: Tag containing the error message.

CID: The command identification defined in the command that generated the response. You can use this value to associate the response with the command that caused the response.

STATUS: If the command was unsuccessful, the value is "ERR".

MESSAGE: Tag containing the error message.

ID: The error's ID number. This ID can be used to perform some action or to convert the error message into text that is customized for your application.

GETSCORES Command

Use the GETSCORES command to get current results of the interview. Each attribute is scored according to the end-user's preferences. Precision Choice uses answers to the questions to determine these scores. The GETSCORES command can be

sent to Precision Choice any time after the rating and importance questions have been generated and answered. The more questions that the end-user answers, the more precise the scores are. The GETSCORES command must be contained inside of a BATCH tag.

Format

<BATCH UID="678" INTERVIEWID="123">
<GETSCORES CID="878"/>
</BATCH>

Elements of the GETSCORES Command

Note: XML is case-sensitive.

GETSCORES: Tag causing Precision Choice to generate and return user preference scores. A value for each attribute and level in the study is returned.

CID: The command identification returned with the response to this command. You can use this value to associate the Precision Choice response with the command that caused it. It is alphanumeric and required. Precision Choice does not use the value, but returns in the response to the command.

Response Format 1 (Successful)

<BATCHRESPONSE UID="678" INTERVIEWID="123">
<CMDRESPONSE CID="878" STATUS="OK">
<SCORES>
<ATTRIBUTE
NAME="cpu"
RELATIVEIMPORTANCE=".5017654">
<LEVEL NAME="PII">
<UTILITY
NAME="FinalScaledUtilities"
VALUE="48">
</LEVEL>
</ATTRIBUTE>
</SCORES>
</CMDRESPONSE>

</BATCHRESPONSE>

Elements of Response Format 1

Note: XML is case-sensitive.

BATCHRESPONSE: Response to a batch command. A BATCHRESPONSE contains a CMDRESPONSE for each command in the batch.

UID: Value matching the UID submitted in the previous corresponding batch tag.

INTERVIEWID: Value matching the INTERVIEWID submitted in the previous corresponding batch tag. If no INTERVIEWID is specified on the batch tag, Precision Choice generates a unique ID and sends it back through this element.

CMDRESPONSE: This tag contains the Precision Choice response to a command.

CID: The command identification defined in the command that generated the response. You can use this value to associate the response with the command that caused the response.

STATUS: If the command was successful, the value is "OK".

SCORES: Tag containing the current scores of the end-user's preferences.

ATTRIBUTE: Tag identifying the attribute and its relative importance to the end-user. There is one ATTRIBUTE tag for every attribute defined in the study.

NAME (for the ATTRIBUTE tag): The attribute's name defined in the study.

RELATIVEIMPORTANCE: Rates how important this attribute is to the end-user. The importance is a decimal number expressed in a scale between 0 and 1, inclusive.

LEVEL: Identifies an attribute's level. It contains the utility score of the level, and there is one LEVEL tag for each level defined in the study.

NAME (for the LEVEL tag): The level's name defined in the study.

UTILITY: Tag identifying the type of the utility and the value. This version of Precision Choice returns only the final scaled utility value.

NAME (for the Utility tag): The utility's name. Currently, the only value is "FinalScaledUtilities."

VALUE: The utility's value expressed as a decimal number.

Response Format 2 (Error)

<BATCHRESPONSE UID="678" INTERVIEWID="123">

<CMDRESPONSE CID="878" STATUS="ERR">

<MESSAGE ID="415">No scores calculated.</MESSAGE>

</CMDRESPONSE>

<BATCHRESPONSE>

Elements of Response Format 2

Note: XML is case-sensitive.

BATCHRESPONSE: Response to a batch command. A BATCHRESPONSE contains a CMDRESPONSE for each command in the batch.

UID: Value matching the UID submitted in the previous corresponding batch tag.

INTERVIEWID: Value matching the INTERVIEWID submitted in the previous corresponding batch tag. If no INTERVIEWID is specified on the batch tag, Precision Choice generates a unique ID and sends it back through this element.

CMDRESPONSE: Tag containing the error message.

CID: The command identification defined in the command that generates the response. You can use this value to associate the response with the command that caused the response.

STATUS: If the command was unsuccessful, the value is "ERR".

MESSAGE: Tag containing the error message.

ID: The error's ID number. This ID can be used to perform an action or to convert the error message into text customized for your application.

GETRECOMMENDATIONS Command

Use the GETRECOMMENDATIONS command to get the product recommendations from Precision Choice. Precision Choice's product recommendations are based upon the current scores of the interview. Before sending this command, the GETSCORES command must have been sent previously to Precision Choice for this interview. The recommendations are based upon the last scores generated for this interview. If you call GETSCORES and the end-user then answers more questions, the

new answers do not effect the product recommendations. For new answers to be considered, GETSCORES would have to be resent.

Executing this command also causes Precision Choice to store the current user preferences for later data analysis. You can specify how many products to return to your application. Repeatedly calling the GETRECOMMENDATIONS command causes Precision Choice to page through the catalog and return the next set of products. For example, assume that there are one hundred products in a catalog. If you send a GETRECOMMENDATIONS command specifying five products, Precision Choice returns the top five products. If you send another GETRECOMMENDATIONS command specifying five products, Precision Choice returns the next five products. You cannot move back up the list, so your application must remember any previous products that were returned, if your application requires that type of functionality. The product recommendations include all information associated with the product in the catalog. If a product contains attributes not defined in the study, the value for each of those attributes is -1.

Format of the GETRECOMMENDATIONS Command

```
<BATCH UID="678" INTERVIEWID="123">
<GETRECOMENDATIONS CID="878" MAX="5"/>
</BATCH>
```

Elements of the GETRECOMMENDATIONS Command

Note: XML is case-sensitive.

GETRECOMMENDATIONS: Tag that causes Precision Choice to generate and return product recommendations.

CID: The command identification returned with the response to this command. You can use this value to associate the Precision Choice response with the command that caused it. It is alphanumeric and required. Precision Choice returns the value in the response to the command.

MAX: Allows you to specify the maximum number of products to return. Precision Choice returns up to this number of products.

Response Format 1 (Successful)

```
<BATCHRESPONSE UID="678" INTERVIEWID="123">
<CMDRESPONSE CID="878" STATUS="OK">
<MAXPRODUCTS VALUE="500"/>
<PRODUCTS>
<PRODUCT
ID="1"
NAME="Compaq Presario 1900"
FIT="77"
RANK="4" >
<ATTRIBUTE
ID="Brand"
VALUE="Compaq"
FIT="55
RELATIVEIMPORTANCE="60"/>
<ATTRIBUTE
ID="Price"
VALUE="2950"
FIT="55
RELATIVEIMPORTANCE="60"/>
<ATTRIBUTE
ID="RAM"
VALUE="128"
FIT="55
RELATIVEIMPORTANCE="60"/>
<ATTRIBUTE
ID="Speed"
VALUE="366"
FIT="55
RELATIVEIMPORTANCE="60"/>
<ATTRIBUTE
ID = "Processor"
```

VALUE="Pentium II"
FIT="55
RELATIVEIMPORTANCE="60"/>
<REFERENCES>
<REFERENCE
TYPE="image">someimage.jpg</REFERENCE>
<REFERENCE
TYPE="details">inspiron7000.html</REFERENCE>
</REFERENCES>
</PRODUCT>
</PRODUCTS>
</CMDRESPONSE>
</BATCHRESPONSE>

Elements of Response Format 1

Note: XML is case-sensitive.

BATCHRESPONSE: Response to a batch command. A BATCHRESPONSE contains a CMDRESPONSE for each command in the batch.

UID: Value matching the UID submitted in the previous corresponding batch tag.

INTERVIEWID: Value matching the INTERVIEWID submitted in the previous corresponding batch tag. If no INTERVIEWID is specified on the batch tag, Precision Choice generates a unique ID and sends it back through this element.

CMDRESPONSE: Tag containing product recommendations for the interview.

CID: The command identification defined in the command that generates the response. You can use this value to associate the response with the command that caused the response.

STATUS: If the command was successful, the value is "OK".

MAXPRODUCTS: Tag containing the total number of products in the catalog.

VALUE (for the MAXPRODUCTS tag): Total number of products in the catalog.

PRODUCTS: Tag containing the set of recommended products.

PRODUCT: Tag defining a product. All attributes and references for one product are defined in this element, as well as a text name of the product. This value can be displayed to the end-user.

ID (for the PRODUCT tag): The product's key. It is defined in the catalog and must be unique.

NAME: The product's name defined in the catalog.

FIT: Value expressing how closely the product matches the user's preferences, expressed as a decimal number. The larger the number, the better the fit. The range of values is 0 to 100, inclusive.

RANK: Tag defining the rank of the product, compared to all other products in the catalog. It is expressed on a scale of 1 to the maximum number of products in the database: 1 is the highest-ranked product.

ATTRIBUTE: A defined attribute of the product. This is also defined in the catalog definition.

ID (for the ATTRIBUTE tag): Defines an attribute of the product. The value is the attribute's name.

VALUE (for the ATTRIBUTE tag): Returns the value of the attribute for this product.

FIT: Value expresses as a decimal number how close this attribute of this product matches the user's preferences. The larger the number, the better the fit. The range of values is 0 to 100, inclusive. If this attribute is not defined in the study, the value is -1.

RELATIVEIMPORTANCE: Rates how important this attribute of a product is to the user, expressed in a scale between 0 and 100, inclusive.

REFERENCES: Optional tag containing all references for a product.

REFERENCE: Optional tag defining a reference for a product: an HTML page, a JPEG file, or anything else you would like to associate with this product. You can only use this tag can in a REFERENCES tag. Precision Choice does not use this information, but returns it in the response to the GETRECOMMENDATIONS command.

TYPE: Defines the type of reference. Precision Choice does not use it, but it is available for your application to base processing on.

Response Format 2 (Error)

<BATCHRESPONSE UID="678" INTERVIEWID="123">

<CMDRESPONSE CID="878" STATUS="ERR">

<MESSAGE ID="415">Invalid answer.</MESSAGE>

</CMDRESPONSE>

<BATCHRESPONSE>

Elements of Response Format 2

Note: XML is case-sensitive.

BATCHRESPONSE: Response to a batch command. A BATCHRESPONSE contains a CMDRESPONSE for each command in the batch.

UID: Value matching the UID submitted in the previous corresponding batch tag.

INTERVIEWID: Value matching the INTERVIEWID submitted in the previous corresponding batch tag. If no INTERVIEWID is specified on the batch tag, Precision Choice generates a unique ID and sends it back through this element.

CMDRESPONSE: Tag containing the error message.

CID: The command identification defined in the command that generated the response. You can use this value to associate the response with the command that caused the response.

STATUS: If the command was unsuccessful, the value is "ERR".

MESSAGE: Tag containing the error message.

ID: The error's ID number. This ID can be used to perform some action or to convert the error message into text customized for your application.

ENDINTERVIEW Command

Use the ENDINTERVIEW command to instruct Precision Choice to end the interview. All resources this interview uses are released. If this command is not sent, resources are not be recovered until the time-out value becomes active.

Format

<BATCH UID="678" INTERVIEWID="123">

<ENDINTERVIEW CID="555"/>

</BATCH>

Elements

ENDINTERVIEW: Tag that causes Precision Choice to end the interview and recover all resources this interview consumed.

CID: The command identification returned with the response to this command. You can use this value to associate the Precision Choice response with the command that caused it. It is alphanumeric and required. Precision Choice does not use the value, but returns it in the response to the command.

Response Format 1 (Successful)

<BATCHRESPONSE UID="678" INTERVIEWID="123">

<CMDRESPONSE CID="3" STATUS="OK"/>

</BATCHRESPONSE>

Elements of Response Format 1

Note: XML is case-sensitive.

BATCHRESPONSE: Response to a batch command. A BATCHRESPONSE contains a CMDRESPONSE for each command in the batch.

UID: Value matching the UID submitted in the previous corresponding batch tag.

INTERVIEWID: Value matching the INTERVIEWID submitted in the previous corresponding batch tag. If no INTERVIEWID is specified on the batch tag, Precision Choice generates a unique ID and sends it back through this element.

CMDRESPONSE: This tag contains the Precision Choice response to a command.

CID: The command identification defined in the command that generated the response. You can use this value to associate the response with the command that caused the response.

STATUS: If the command was successful, the value is "OK".

Response Format 2 (Error)

<BATCHRESPONSE UID="678" INTERVIEWID="123">

<CMDRESPONSE CID="878" STATUS="ERR">

<MESSAGE ID="412">Interview not found.</MESSAGE>

</CMDRESPONSE>

<BATCHRESPONSE>

Elements of Response Format 2

Note: XML is case-sensitive.

BATCHRESPONSE: Response to a batch command. A BATCHRESPONSE contains a CMDRESPONSE for each command in the batch.

UID: Value matching the UID submitted in the previous corresponding batch tag.

INTERVIEWID: Value matching the INTERVIEWID submitted in the previous corresponding batch tag. If no INTERVIEWID is specified on the batch tag, Precision Choice generates a unique ID and sends it back through this element.

CMDRESPONSE: Tag containing the error message.

CID: The command identification defined in the command that generated the response. You can use this value to associate the response with the command that caused the response.

STATUS: If the command was unsuccessful, the value is "ERR".

MESSAGE: Tag containing the error message.

ID: The error's id number. This ID can be used to perform an action or to convert the error message into text customized for your application.

Precision Choice Adaptor and Load Balancing

High-Level View of Precision Choice's Engine

Precision Choice's engine is the piece of the Precision Choice solution that processes the interview:

- Generates questions
- Analyzes responses
- Calculates consumer profile data
- Generates product recommendations

As FIG. 21 shows, multiple Precision Choice engines **2110** are grouped into what are referred to as clusters **2105a, 2105b**. Essentially, a cluster is a group of Precision Choice engines. Also, each cluster has an adaptor **2120a, 2120b** that serves as

the entry point, i.e., gateway, into the cluster for merchants' external systems **2130a, 2130b, 2130c**. A cluster **2105a, 2105b** is a group of Precision Choice engines **2110** that share an adaptor **2120a, 2120b** as the entry point. There may be many clusters, each with its own set of Precision Choice engines and its own unique adaptor.

There are three additional points about clusters worth noting:

Each cluster may itself be distributed across process and machine boundaries, i.e., there is no limitation on the physical location of the Precision Choice engines or the adaptor with respect to each other. They may be collocated on the same machine or on separate machines that are networked together. This capability to harness the computing power and resources of multiple physical machines allows Precision Choice to achieve a high level of scalability.

Each engine in the cluster can handle multiple interviews simultaneously.

Each client may have multiple connections with the adaptor. Each connection represents a separate thread of execution, which allows a single client efficiently to conduct multiple simultaneous interviews. Although multiple connections are not required for a client to be able to conduct multiple interviews, using a single connection would probably degrade response time.

The following sections provide details about cluster architecture, beginning with a detailed discussion of the adaptor. There is a discussion of its purpose, its internal architecture, and implementation details for an adaptor, including a case study implementation of an adaptor. Using this knowledge there is a final discussion on the intricacies of load balancing and the cluster concept as they apply to the Precision Choice system.

What is an Adaptor 2120a, 2120b?

An adaptor is a software application that helps customers integrate their system with the Precision Choice engine. Although it is part of the Precision Choice solution, the adaptor is not a part of the core Precision Choice engine. The Precision Choice adaptor serves the following purposes:

1. Acts as a coupling agent when there is a communications protocol mismatch between the client system and Precision Choice's engine.

2. Provides load balancing for multiple Precision Choice engines. This is the minimal function any customer needs from an adaptor, even when there is no protocol mismatch. As detailed below, Precision Choice implements load balancing through clustering.
3. Serves as the gateway to a cluster.

Detailed Structure of the Adaptor

Two Sides of the Adaptor

Conceptually, the adaptor **2210** may be seen as consisting of two sides, as shown in FIG. 22: a Client side **2230** and a Precision Choice side **2240**.

Precision Choice uses RMI as its communications protocol. Client systems **2220** may or may not use RMI. For example, client systems may be CORBA-based or Microsoft DCOM-based. Regardless of the protocol the client system uses, it needs to be able to communicate with Precision Choice.

Precision Choice enables this communication through use of the adaptor **2210**. The adaptor allows Precision Choice to be client-neutral and provides maximum integration flexibility. The design of the Client side varies, depending on the protocol the client uses. The Client side of the adaptor is custom-developed and, depending upon the client system, can vary from very straightforward to extremely complex. The Precision Choice side of the adaptor is fixed, i.e., is the same for all merchants. Internally, the Client side and the Precision Choice side may communicate via any appropriate method.

The Client side of the adaptor receives interview data from the client system, transforms the data into the appropriate format, and sends it to the Precision Choice side of the adaptor. The Precision Choice side of the adaptor sends the interview data to the Precision Choice engine. The engine processes the interview data and sends results back to the adaptor, which are then sent back to the client making the call. This is a synchronous operation, i.e., the client thread sending interview data to the adaptor is blocked until the adaptor returns the results. A sample scenario follows the discussion of how the commands that a client system sends are routed to the appropriate engine for processing.

The Adaptor's Routing of Commands

As stated above, the adaptor routes commands that contain interview data to the Precision Choice engine. As FIG. 21 illustrates, there may be many engines linked to a single adaptor, and the adaptor must determine, with minimal decoding, which engine should receive the command.

As discussed above, there are two broad categories of commands: batch and non-batch. Based on the type of command and with minimal decoding, the adaptor decides where to send the command.

Routing Non-Batch Commands

GETSTATUS Command

The GETSTATUS command is sent to every engine registered with the adaptor. The adaptor assimilates responses from all engines into one response and sends the response out to the client system.

For example, assume that two Precision Choice engines are registered with the adaptor, i.e. the cluster has two Precision Choice engines. When the adaptor receives a GETSTATUS command, it sends it to both Precision Choice engines.

Assume that one Precision Choice engine sent the following response:

<CMDRESPONSE cid="100" status="OK">
<FREEMEMORY value="556789"/>
<TOTALINTERVIEWS value="120"/>
</CMDRESPONSE>

Assume that the other Precision Choice engine sent the following response:

<CMDRESPONSE cid="100" status="OK">
<FREEMEMORY value="234845"/>
<TOTALINTERVIEWS value="122"/>
</CMDRESPONSE>

The adaptor would send the following response to the client system:

<CMDRESPONSE cid="100" status="OK">
<FREEMEMORY value="791634"/>
<TOTALINTERVIEWS value="242"/>
</CMDRESPONSE>

The adaptor sums the free memory and the total interviews from each engine and sends a combined response for that cluster back to the client system.

SETSTUDY Command

This command is sent to any available engine registered with the adaptor.

SETCATALOG Command

This command is sent to any available engine registered with the adaptor.

Routing Batch Commands

For batch commands, there are two cases to consider, based on whether the batch contains an INTERVIEWID.

Case 1: Batch does not have an INTERVIEWID

A batch with no INTERVIEWID is a batch for a new interview. The adaptor detects this and creates a new INTERVIEWID. This

INTERVIEWID is based on the UID the client supplies as part of the. The INTERVIEWID must be unique. Examples of creating a unique INTERVIEWID based on the UID include the following:

UID + timestamp

UID + Globally Unique Identifier

After creating this INTERVIEWID, the adaptor uses its internal load balancing algorithm **2250** to select an available engine and sends the batch to this engine. The adaptor also "remembers" where it sent this interview and sends all subsequent batches for this interview to the same engine. The adaptor returns this INTERVIEWID as part of the response to the batch as described more fully above. The client is responsible for saving this

INTERVIEWID and including it in every subsequent batch submitted for that interview.

Case 2: Batch has an INTERVIEWID

As discussed above, based on the INTERVIEWID, the adaptor sends the interview to the appropriate Precision Choice engine. The adaptor tracks which engine received the batch the first time. Each new batch with the same INTERVIEWID is routed to the same engine because the engine maintains a state for each interview. This

means that each Precision Choice engine is a "sticky server," and the client must send an interview to the same adaptor each time.

An Example of the Data Flow Sequence

FIG. 23 shows two client systems, a cluster with three engines, and an adaptor. FIG. 24 shows Client System 1 starting a new interview by sending a STARTINTERVIEW command. Since this is a new interview, the batch does not have an INTERVIEWID. The adaptor creates an INTERVIEWID for this interview and assigns this interview to Precision Choice Engine 1. The adaptor uses its internal load balancing algorithm to select this engine.

FIG. 25 shows Client System 2 starting a new interview by sending a STARTINTERVIEW command. Since this is a new interview, the batch does not have an INTERVIEWID. The adaptor creates an INTERVIEWID for this interview and assigns the interview to Precision Choice Engine 2. The adaptor uses its internal load balancing algorithm to select this engine.

FIG. 26 shows client Client System 2 sending a GETNEXTQUESTIONSET command to the adaptor. This command is sent in a batch with the INTERVIEWID the adaptor generates and returns as part of starting the interview. The adaptor recognizes the INTERVIEWID and forwards the command to Precision Choice Engine 2, based on this INTERVIEWID.

FIG. 27 shows Client System 1 sending a GETNEXTQUESTIONSET command to the adaptor. This command is sent in a batch with the INTERVIEWID generated and returned by the adaptor as part of starting the interview. The adaptor recognizes the INTERVIEWID and forwards the command to Precision Choice Engine 1, based on this INTERVIEWID.

FIG. 28 shows Client System 1 sending an ENDINTERVIEW command to the adaptor. This command is sent in a batch with the INTERVIEWID generated and returned by the adaptor as part of starting the interview. The adaptor recognizes the INTERVIEWID and forwards the command to Precision Choice Engine 1, based on this INTERVIEWID. As a result of this command, the engine removes all state for this interview and informs the adaptor of the interview's end through the fire method of the

CPEvents interface. The adaptor no longer recognizes any interviews with this INTERVIEWID.

FIG. 29 shows Client System 2 sending an ENDINTERVIEW command to the adaptor. This command is sent in a batch with the INTERVIEWID the adaptor generates and returns as part of starting the interview. The adaptor recognizes the INTERVIEWID and forwards the command to Precision Choice Engine 2, based on this INTERVIEWID. As a result of this command, the engine removes all state for this interview and informs the adaptor of the interview's end through the fire method of the CPEvents interface. The adaptor no longer recognizes any interviews with this INTERVIEWID.

Implementing an Adaptor

There are two components of an adaptor. These are represented as the Client side **2230** and the Precision Choice side **2240** in FIG. 22. The Client side illustrates the connection to the merchant's system; this part of the adaptor varies by project and merchant. The Precision Choice side of the adaptor illustrates the connection to Precision Choice: this part of the adaptor is fixed. Online Insight Client Services personnel should be able to copy the code for the Precision Choice side and use it without any changes.

The Precision Choice Side 2240

This side of the adaptor is fixed since the Precision Choice engine protocol is fixed and should not require any implementation changes from what is already provided. It is useful to know what it would take to implement the Precision Choice side to understand the Precision Choice solution fully.

There are two interfaces that must be implemented for the Precision Choice side of the adaptor:

CommandProcessRegister

CPEvents

CommandProcessRegister Interface

The CommandProcessRegister interface has two methods. The interface definition is included below for reference.

package com.onlineinsight.intercomponent.adaptor;

```
import com.onlineinsight.intercomponent.CommandProcessor.*;

import java.rmi.RemoteException;
import java.rmi.Remote
;
/**
* The CommandProcessorRegister interface is used by the
* CommandProcessor to register with an Adaptor when it
* starts and to unregister when it shuts down.
*/

public interface CommandProcessorRegister extends Remote {

/**
* The register method is used to register Command Processors
* with the Adaptor. The Adaptor is responsible for load
* balancing between the Command Processors. Each Command
* Processor is  responsible for finding and registering with
* the proper Adaptor.
* @param cmdProcessor - the processor that is registering
* with the adaptor @exception AdaptorException,
* RemoteException
*/
public void register(CommandProcessor cmdProcessor) throws
AdaptorException,RemoteException;

/**
* The unregister method is used to unregister a Command
* Processor.
* @param cmdProcessor - the processor that is unregistering * with the adaptor
@exception AdaptorException,
```

```
        * RemoteException
        */
        public void unregister(CommandProcessor cmdProcessor) throws
AdaptorException,RemoteException;

        }
```

Register Method

An engine can register with the adaptor by calling the Register method. An engine should only register with one adaptor. Every adaptor resides on a single machine with a unique host name, and on that machine each adaptor has a unique name and port. The adaptor namespace has the following form:

Adaptor namespace = machine name + port (for a machine) + adaptor name

This namespace allows an engine to find/identify the adaptor uniquely in a distributed system.

Unregister Method

The Unregister method allows an engine to remove itself from a cluster.

CPEvents Interface

The CPEvents interface has one method. The interface definition is included below for reference.

```
        package com.onlineinsight.intercomponent.CommandProcessor;

        import java.rmi.Remote;
        import java.rmi.RemoteException;

        public interface CPEvents extends Remote
        {
        public void fire(CommandProcessorEvent e) throws
        RemoteException;
        }
```

Fire Method

An engine calls the fire method to inform systems external to the engine, such as the adaptor, of events that these external systems express "interest" in. For example, the adaptor is interested in knowing when an interview ends. An interview can end voluntarily or as the result of a time-out. The adaptor needs to know when an interview ends because it maintains state for each interview. When an interview ends, this state needs to be removed. If this state is not removed, it would lead to excessive state buildup over time, ultimately dominating all server resources and bringing the server "down." During startup, the engine automatically registers the adaptor to receive "End Interview" events.

When an interview ends (voluntarily or because of a time-out), the engine calls the Fire method to notify the adaptor that the interview ended. As a parameter, this method gets an event that contains the INTERVIEWID of the interview that has ended. Based on this knowledge, the adaptor may perform all necessary cleanup pertaining to that interview.

A Case Study of A Sample Adaptor

A merchant is a leader in e-commerce web-based applications targeted towards selling to the end consumer (B2C). They have an extensive array of products and services that they offer to other potential customers. Due to the complex nature of these products and services, consumers find it difficult to buy these products and services off the Web. Realizing this, the merchant decided to invest heavily in a Precision Choice-centric solution.

The merchant has a CORBA-based architectural framework in which they have invested a lot of time, money, and energy. Online Insight developed a solution that leverages their existing infrastructure by means of our adaptor architecture.

The Adaptor Design

The merchant uses CORBA, specifically Inprise's Visibroker ORB. They have two interfaces that all third parties interfacing with their system must implement. They provided those interface definitions.

The merchant sent all commands in the XML format Precision Choice requires.

The merchant requires the ability to shutdown an entire cluster on-demand.

The merchant requires the ability to add engines to a cluster dynamically.

The merchant requires a robust load balancing solution.

We met items 4 and 5 with the nature of the Precision Choice solution. The sections on Clusters and Load Balancing above provide details. We support items 1 and 3 through the adaptor. If the merchant had not sent all commands in the XML format Precision Choice requires (item 2), Online Insight would have had to translate whatever format they provided into the correct XML format for the Precision Choice engine.

The IDL the merchant provided as part of item 1 is shown below:

```
module com
{
module ACME
{
exception InvalidCommandSequence
{
};

interface User
{
string submit(in string commandXML) raises
(InvalidCommandSequence);
};

interface Manager
{
string submit(in string commandXML) raises
(InvalidCommandSequence);
};
};
};
```

As a result of item 1, in this case the Client side of the adaptor becomes a CORBA server with at least two CORBA objects; one object implements each interface. Since Online Insight does not care which interface the merchant uses to send

us the XML commands, we simply funnel all XML commands from either object to the Precision Choice side of the adaptor. Since the command XML we receive is already in the proper format for the engine, no intermediate processing is required.

To satisfy item 3, we implemented another CORBA object with a new CORBA interface. This interface is shown in the IDL code below:

```
module com
{
module onlineinsight
{
module adaptor
{
interface ClusterController
{
void shutdown();
};
};
};
};
```

The client calls the shutdown method of the object implementing this interface to shutdown the cluster.

The following points about this adaptor solution are noteworthy:

The client is able to bind to the CORBA objects with Visibroker's proprietary "bind" method. The merchant is already comfortable with our solution. The adaptor supports configuration through a properties file or command line parameters to set up names for the two CORBA objects implementing their required interfaces and to set up the name for the CORBA object implementing the shutdown interface. Due to Visibroker requirements, the names for these CORBA objects must be unique within the merchant's domain.

The client is notified of errors in actual command sequences by the use of CORBA user exceptions, as the merchant requires. Any errors in processing allowed

command sequences are communicated through the XML responses for each command itself.

FIG. 30 illustrates the process described above.

Overview of Load Balancing

The Problem of Load Balancing

Enterprise applications have to worry about the ratio of clients to server resources in a way that traditional desktop applications never did. Typical desktop applications were limited to the number of clients that used them simultaneously. The advent of the Web broke all these boundaries and has resulted in a new class of Web applications subject to hundreds of thousands, even millions, of clients simultaneously. Just think of how many people make stock trades at E*TRADE on a busy day. Although tricky and complex, solving load balancing problems is not impossible.

Any system, no matter how well-designed, has a maximum amount of "load" that it can handle while maintaining adequate performance. Load refers to work the system is being asked to perform; adequate performance refers to a user-tolerable response time. Systems that are not well-designed may fail under maximum or near-maximum load levels. No matter how well-designed a system is, there is always a maximum load that it can handle. Being able to "balance the load" across multiple similar systems has become increasingly important.

The simplest solution to the load balancing problem is "static" load balancing: a fixed number of "servers," i.e., systems or web-applications, handle incoming requests from clients. Such systems traditionally suffer from a number of problems: limited scalability, uneven load distribution resulting in wasted computing bandwidth, and unacceptable fault-tolerance. Online Insight is familiar with these problems and has not developed such an implementation. We have adopted a much more flexible and robust approach, namely a dynamic and configurable load-balancing approach, for our Precision Choice solution.

A need exists for Web-enabled, enterprise applications to handle a substantial load without experiencing a noticeable degradation of performance. We incorporated several key load balancing features into the design of the Precision Choice solution.

As explained above, load balancing in the Precision Choice solution revolves around the concept of a cluster.

What is a Cluster?

A cluster is the smallest divisible unit for load balancing. Online Insight's clusters are not the same as clusters described in some operating systems/middleware systems such as COM+. The clusters referred to in this document apply only to the Precision Choice solution.

A cluster consists of one adaptor, the gateway to the cluster, and any number of Precision Choice engines. The client controls the number of Precision Choice engines joining a cluster by configuring a Precision Choice engine to register with a particular adaptor (see details above). Adaptors never reject requests from Precision Choice engines to join a cluster, and an engine may unregister any time by calling the unregister method of the CommandProcessorRegister interface, as discussed in the Adaptor section (above).

A client creates a new cluster by starting a new adaptor and then starting engines that register with the adaptor: an empty cluster with one adaptor and no engines would serve no useful purpose. Precision Choice allows an unlimited number of clusters. Clients may control load balancing only between clusters, i.e. clients cannot influence any decisions within a cluster as to which engine the system may use. This decision is at the sole discretion of the adaptor, which makes it by using the load balancing algorithm implemented within the adaptor. The following section discusses the loads balancing algorithm in more detail.

There is no standard way to control a cluster, such as shutting it down. The method of control is client-specific. For example, in the case study discussed earlier, Online Insight implemented a CORBA object with a cluster control interface that has a shutdown method. The client used the shutdown method of the cluster control interface to shutdown a cluster. Online Insight provided no way to shutdown individual engines in a cluster, only the entire cluster. The client has the capability of adding new engines to the cluster (a standard feature discussed above), but no control over an individual engine after it is added to a cluster. The only control the client has is to shutdown the entire cluster.

The Load Balancing Algorithm

As discussed above, the adaptor retains a map of the INTERVIEWID and the Precision Choice engine processing the interview corresponding to that INTERVIEWID. The Precision Choice solution is based on a sticky server implementation: once an engine is used for an interview, the same engine must be used for the life of that interview. Each engine maintains state corresponding to each interview being conducted using that engine. When a new interview starts, the adaptor must select an engine to process that interview. The adaptor uses the load balancing algorithm to select an engine.

This default load balancing algorithm implements round robin scheduling. Through customization, this algorithm could be replaced with one that evaluates the state of the engines and sends an interview to the most appropriate engine.

Assume that there is a cluster with three engines (A, B, and C) and three active interviews (1, 2, and 3):

Using round-robin scheduling, the adaptor sends interview 1 to Engine A. It then sends interview 2 to Engine B and interview 3 to Engine C. To understand how the load-balancing algorithm works, assume that interview 3 is aborted (see the Table below). The round robin algorithm does not account for this and continues to assign interviews to the engines sequentially. When a new interview (interview 4) needs to be assigned, the adaptor sends it to Engine A, the next engine in the sequence. The adaptor does not evaluate the load on the engines before making the assignment. In the Round Robin column in the Table below, Engine A has interviews 1 and 4, while Engine 3 has no interviews.

Another algorithm could evaluate the system before assigning the interview. When the algorithm finds that Engine C has no interview, it would assign the new interview there, i.e., to the engine with the lightest load. In the Alternate column in the Table below, the algorithm assigns interview 4 to Engine C.

The following Table summarizes these approaches for assigning interviews to engines:

Engine	Interview 3 Aborted	Round Robin	Alternate

A	1	1 and 4	1
B	2	2	2
C	None	None	4

Recommendations for Load Balancing

Although Online Insight has effectively balanced the engines within a cluster, the adaptor for each cluster is not load balanced in that cluster itself. Stated differently, no matter how many engines there are in a cluster, there is only one adaptor in that cluster, serving as the sole entry point for that cluster. It is possible that the adaptor itself may become a bottleneck under high-load situations.

The processing an adaptor does is extremely small when compared to the processing each engine does during the course of an interview. The possibility of the adaptor actually becoming a bottleneck is smaller than that of an individual engine becoming the bottleneck. When a cluster contains a large number of engines, all of which are used fairly heavily, the processing the adaptor does may become noticeable. There should not be too many engines in a cluster. Online Insight will determine the ideal number of engines for a cluster based on experience.

Precision Choice Configuration

The foregoing discussion covers the settings or properties that you can configure for the Precision Choice engine. These properties are considered static since the engine does not recognize the changes at runtime, i.e., these properties are read in when the engine (or components of the engine) starts and used until the engine is shutdown.

The components of the engine are the CommandProcessor, Datastore, Interviewer, ConjointEngine, and an Adaptor. These components form the Precision Choice engine. These components rely on configuration information contained in properties files and XML files.

Definitions

Port

A port refers to a TCP/IP port number. The TCP/IP protocol uses these numbers as endpoints in a communication link between (in this case) components. The numbers range between 0 and 65536. Ports between 0 and 1023 are reserved as Well-Known

Port Numbers: these ports are reserved and should not be used by any Precision Choice component. See RFC 1700 for more details on Well-Known Port Numbers.

Absolute File Name

An Absolute File Name is the name of file that includes the path to the name. For example, /pc/conf/precisionchoice.properties is an absolute file name since it includes the path to the file. By contrast, precisionchoice.properties is a relative file name since it does not include the path to the file.

#

The pound sign (#) is present at the beginning of any line in a properties file that is a comment. The component using the file does not process the comment.

CommandProcessor Properties

General Properties

CommandPort

Description The unique TCP port a given Command Processor uses. Each Command Processor should have a unique command port.

Valid Values An unused port number on the machine the Command Processor runs on. Ports between 0 and 1023 are reserved as Well-Known Port Numbers: these ports are reserved and should not be used by any Precision Choice component. See RFC 1700 for more details on Well-Known Port Numbers.

Example CommandPort = 2000

PCpropertiesFile

Description

The full path of the precisionchoice.properties file that contains information specific to the Precision Choice recommendation engine, such as license keys, units, and other configuration information. Paths must use the Unix style forward slash (/).

Valid Values The absolute file name of the Precision Choice properties file.

Example PCpropertiesFile = C:/pc/conf/precisionchoice.properties

ErrorLog

Description

The full path of a file used to log the Command Processor's system errors. If this property is blank, error logging is dumped on the screen. Paths must use the Unix style forward slash (/).

Valid Values The absolute file name of the desired error log file.

Example ErrorLog = c:/pc/log/error.log

Internal Components of the Command Processor

Components

Description A comma-delimited list of internal Command Processor components that require bootstrapping as RMI servers.

Example Components = StatusManager,EventManager,Commander

Communicator

Description A fully qualified Java class name of the Command Processor's internal Communicator component

Valid Values com.onlineinsight.CommandProcessor.Communicator.Communicator

Example Communicator = com.onlineinsight.CommandProcessor.Communicator.Communicator

Commander

Description A fully qualified Java class name of the Command Processor's internal Commander component

Valid Values com.onlineinsight.CommandProcessor.Commander.CommanderImpl

Example Commander = com.onlineinsight.CommandProcessor.Commander.CommanderImpl

StateManager

Description A fully qualified Java class name of the Command Processor's internal State Manager component

Valid Values com.onlineinsight.CommandProcessor.StateManager.StateManager

Example StateManager = com.onlineinsight.CommandProcessor.StateManager.StateManager

StatusManager

Description A fully qualified Java class name of the Command Processor's internal Status Manager component

Valid Values com.onlineinsight.CommandProcessor.StatusManager.StatusManagerImpl

Example StatusManager = com.onlineinsight.CommandProcessor.StatusManager.StatusManagerImpl

EventManager

Description A fully qualified Java class name of the Command Processor's internal Event Manager component

Valid Values com.onlineinsight.CommandProcessor.EventManager.EventManager

Example EventManager = com.onlineinsight.CommandProcessor.EventManager.EventManager

State Manager Properties

StateTimeout

Description The number of minutes between interactions that causes the State Manager to timeout an interview. When an interview is timed out, the State Manager removes it, and it is no longer valid.

Valid Values A number greater than or equal to 1.

Example StateTimeout = 20

GarbageCollectionFreq

Description The frequency in minutes that the State Manager's garbage collector thread is invoked to remove timed-out interviews from the State Manager. This garbage collector is not related to the JVM garbage collector or Java's garbage collection scheme. The State Manager's garbage collector is used to removed timed-out interviews.

Valid Values A number greater than or equal to 1.

Example GarbageCollectionFreq = 10

Commander Properties

XMLClassMapping

Description The full path of the Commander.properties file, containing a mapping between valid XML tag names and the corresponding fully qualified Java class names of the Command Objects that perform appropriate actions. Paths must use the Unix style forward slash

(/).

Valid Values The absolute file name to a file that contains the Commander properties.

Example XMLClassMapping = E:/pc/conf/Commander.properties

XMLParser

Description The Commander uses a standard XML parser to parse and process commands. The XML for these commands have a DTD that defines the correct form for the commands. The Commander can be set up to use either a validating parser or a non-validating parser. If the validating parser is used, any XML command that does not conform to the DTD is not processed. Otherwise, the Commander attempts to process the command, even though it may not be well-formed. This setting affects the speed at which the Commander can process commands. If a validating parser is used, parsing of the command is slower and, therefore, the processing and execution of the command is slower. It is good to use the validating parser when the system is first setup to test that well-formed commands are being sent to the system. Once it has been

established that the Commander is consistently receiving well-formed commands, it makes sense to use the non-validating parser.

Valid Values com.sun.xml.parser.Parser

com.sun.xml.parser.ValidatingParser

Example XMLParser = com.sun.xml.parser.Parser

Adaptor Properties

AdaptorName

Description Adaptor with which the Command Processor registers.

Valid Values A string containing the Adaptor's name.

Example AdaptorName = adaptor

AdaptorHost

Description The name or IP address of the machine running the Adaptor.

Example AdaptorHost = 10.1.1.24

AdaptorPort

Description The port though which the Adaptor and a given Command Processor communicate.

Valid Values An unused port number on the machine running the Adaptor. Ports between 0 and 1023 are reserved as Well-Known Port Numbers: these ports are reserved and should not be used by any Precision Choice component. See RFC 1700 for more details on Well-Known Port Numbers.

Example AdaptorPort = 9999

Datastore Properties

DatastoreName

Description The name of the Datastore that a given Command Processor queries for study and catalog (product) information.

Valid Values A string containing the Datastore's name.

Example DatastoreName = PC2

DatastoreHost

Description The name or IP address of the machine running the Datastore.

Example DatastoreHost = 10.1.1.100

DatastorePort

Description The port through which the Datastore and a given Command Processor communicate

Valid Values An unused port number on the machine running the Datastore. Ports between 0 and 1023 are reserved as Well-Known Port Numbers: these ports are reserved and should not be used by any Precision Choice component. See RFC 1700 for more details on Well-Known Port Numbers.

Example DatastorePort = 1299

Precision Choice Properties

The Precision Choice property file contains settings or properties for two internal components of the Precision Choice engine: the Interviewer and the ConjointEngine components. The Interviewer component is responsible for generating questions and processing the responses to the questions. The ConjointEngine is responsible for calculating conjoint utilities during the interview. The name of this property file is specified in the CommandProcessor.properties file with the key of PCpropertiesFile.

Interviewer Properties

LicenseKey1, LicenseKey2

Description These properties are license codes that Online Insight's licensing program generates. These codes are used, along with other keys embedded in the software, to verify at runtime that a licensed edition of the software is running.

Valid Values Generated values from Online Insight's license program.

Example LicenseKey1=472780325

LicenseKey2=976597200000

com.onlineinsight.study.unit.xmlfile

Description This setting points to the unit file. The unit file details the measurement units and display formats for the interview. Paths must use the UNIX style forward slash (/).

Valid Values The XML file's name of the unit with absolute path, i.e., /pc/conf/units.xml

Example com.onlineinsight.study.unit.xmlfile=c:/pc/conf/units.xml

com.onlineinsight.interviewer.xmlfile

Description This property sets the XML file that the engine uses to setup the interview. Paths must use the UNIX style forward slash (/).

Valid Values The name of the unit XML file with absolute path, i.e., /pc/conf/interview.xml

Example com.onlineinsight.interviewer.xmlfile = c:/pc/conf/interview.xml

com.onlineinsight.interviewer.
validator

Description This property tells the interviewer which validator to use. A validator ensures that the study and catalog are valid to use together in an interview.

Valid Values A fully qualified name of a class that implements the com.onlineinsight.interviewer.InterviewValidator interface. Any class used to validate the study and catalog must be visible to the interviewer component. The safest way to ensure that the class is visible is to make it public in whatever package it is implemented. Currently, there are two implementations of this interface. The first implementation is the com.

onlineinsight.interviewer.AttributeSubsetValidator class. This class ensures that the attributes of the study are a subset of the attributes of the product attributes in the catalog. The second implementation is the com.onlineinsight.interviewer.NullValidator class that performs no validation and assumes that the study and the catalog can be used together.

Example com.onlineinsight.interviewer.validator=
com.onlineinsight.interviewer.validator.
AttributeSubsetValidator

level_creation_strategy

Description This property sets the name of the class that creates levels for user-specific attributes. User-specific attributes are unique to each user taking an interview. Typically, they are passed in to the engine at the start of the interview. The engine needs to generate levels for each attribute and a class that implements the com.onlineinsight.interviewer.study.LevelCreationStrategy interface.

Valid Values The fully qualified name of a class that implements the com.onlineinsight.interviewer.study.LevelCreationStrategy interface.

Example level_creation_strategy=com.onlineinsight.
interviewer.study.FourEqualLevelStrategy

ConjointEngine Properties

com.onlineinsight.conjointengine.
adaptive.ACAStateImpl.
maxNumberOfAttrsForPairs

Description This property sets the maximum number of attributes to be used in the Pairs section of the interview. Although this property sets the maximum number that the Pairs section uses, there can be many attributes in a study. The engine uses several techniques to limit the attributes used in the Pairs section.

Valid Values Any number greater than 0.

Example com.onlineinsight.conjointengine.adaptive.
ACAStateImpl.maxNumberOfAttrsForPairs = 5

com.onlineinsight.conjointengine.
adaptive.ACAStateImpl.
maxNumberOfLevelsForPairs

Description This property sets the maximum number of levels per attribute to be used in the Pairs section of the interview.

Valid Values 5 is the default setting, but any non-negative integer value can be used.

Example com.onlineinsight.conjointengine.adaptive.
ACAStateImpl.maxNumberOfLevelsForPairs = 5

com.onlineinsight.conjointengine.
adaptive.ACAStateImpl.
minNumberOfLevelsPerAttr

Description This property sets the minimum number of levels per attribute.

Valid Values 2 is the default setting and also the minimum value for this property, but any non-negative integer value can be used.

Example com.onlineinsight.conjointengine.adaptive.
ACAStateImpl.minNumberOfLevelsPerAttr = 2

com.onlineinsight.conjointengine.
adaptive.ACAStateImpl.
minNumberOfAttrsPerStudy

Description This property sets the minimum number of attributes in a study that must be present to conduct the interview. This value is normally set to 2 but can be set to require any number of attributes.

Valid Values This property can be set to any number greater than 2.

Example com.onlineinsight.conjointengine.adaptive.
ACAStateImpl.minNumberOfAttrsPerStudy = 2

com.onlineinsight.conjointengine.
adaptive.ACATypedInputHandlers.
handlerclasses

Description The ConjointEngine component uses typed handlers to process input into the component. This property lists these handlers. These values can be

changed to modify how the conjoint engine handles input or when a new type of input is added.

Valid Values A comma-delimited list of fully qualified class names. This list can span multiple lines by adding '\' to the end of each line. Each class enumerated in this list must implement the com.

onlineinsight.conjointengine.InputHandler interface.

Example com.onlineinsight.conjointengine.adaptive.
ACATypedInputHandlers.handlerclasses = \
com.onlineinsight.conjointengine.adaptive.
ACAStudySettingsHandler, \
com.onlineinsight.conjointengine.adaptive.
ACARankingAttributeSettingsHandler, \
com.onlineinsight.conjointengine.adaptive.
ACARatingAttributeSettingsHandler, \
com.onlineinsight.conjointengine.adaptive.
ACAAPrioriAttributeSettingsHandler, \
com.onlineinsight.conjointengine.adaptive.
ACAAttributesToExcludeFromPairsHandler, \
com.onlineinsight.conjointengine.adaptive.
ACAAttributeImportanceSettingsHandler, \
com.onlineinsight.conjointengine.adaptive.
ACAOmittedLevelsSettingsHandler, \
com.onlineinsight.conjointengine.adaptive.
ACAUnacceptableLevelsSettingsHandler, \
com.onlineinsight.conjointengine.adaptive.
ACAPairQuestionResponseHandler, \
com.onlineinsight.conjointengine.adaptive.
ACACalibrationQuestionResponseHandler

com.onlineinsight.conjointengine.
adaptive.ACATypedOutputHandlers.

handlerclasses

Description The ConjointEngine component uses typed handlers to process and return output to the class that uses the component. These should be changed if the component changes how data is returned to classes that use the ConjointEngine.

Valid Values A comma-delimited list of fully qualified class names. This list can span multiple lines by adding '\' to the end of each line. Each class enumerated in this list must implement the com.

onlineinsight.conjointengine.OutputHandler interface.

Example com.onlineinsight.conjointengine.adaptive.

```
ACATypedOutputHandlers.handlerclasses = \
com.onlineinsight.conjointengine.adaptive.
ACANewStateHandler, \
com.onlineinsight.conjointengine.adaptive.
ACANewPairsQuestionChooserHandler, \
com.onlineinsight.conjointengine.adaptive.
ACAPriorsUtilitiesHandler, \
com.onlineinsight.conjointengine.adaptive.
ACAPriorsUtilitiesForPairsHandler, \
com.onlineinsight.conjointengine.adaptive.
ACAPairsUtilitiesHandler, \
com.onlineinsight.conjointengine.adaptive.
ACAEqualWeightedUtilitiesHandler, \
com.onlineinsight.conjointengine.adaptive.
ACANewCalibrationQuestionChooserHandler, \
com.onlineinsight.conjointengine.adaptive.
ACAFinalWeightedUtilitiesHandler, \
com.onlineinsight.conjointengine.adaptive.
ACACalibrationWeightsHandler, \
com.onlineinsight.conjointengine.adaptive.
ACACalibrationRSquareHandler, \
com.onlineinsight.conjointengine.adaptive.
```

ACARelativeImportanceHandler, \
com.onlineinsight.conjointengine.adaptive.
ACAFinalScaledUtilitiesHandler

com.onlineinsight.conjointengine.
pairsquestionchooserimplclass

Description This property sets the class that determines how pair questions are generated. Essentially, there are three ways to generate pair questions: flex, default, and static. The flex method dynamically selects the attributes and levels for the each pair question. The default method statically selects attributes and dynamically chooses the levels to use within the attributes. The static method statically selects attributes and statically selects the levels within the attributes.

The attributes this method uses are specified in a Pair Attributes Setting XML file specified by the com.
onlineinsight.conjointengine.adaptive.
ACADefaultPairsQuestionChooserImpl.
pairsattributesettingsxml property.
Valid Values com.onlineinsight.conjointengine.adaptive.
ACAFlexNumberPairsQuestionChooserImpl
com.onlineinsight.conjointengine.adaptive.
ACADefaultPairsQuestionChooserImpl
com.onlineinsight.conjointengine.adaptive.
ACAStaticPairsQuestionChooserImpl

or a fully qualified name of a class that implements the com.onlineinsight.intercomponent.conjointengine.
PairsQuestionChooser interface.
Example com.onlineinsight.conjointengine.
pairsquestionchooserimplclass =
com.onlineinsight.conjointengine.adaptive.
ACADefaultPairsQuestionChooserImpl

com.onlineinsight.conjointengine.
adaptive.ACADefaultPairsQuestion
ChooserImpl.maxNumberOfPairs

Description This property is not currently used. The number of pairs questions is set to 7.

Example com.onlineinsight.conjointengine.adaptive.
ACADefaultPairsQuestionChooserImpl.maxNumberOfPairs = 7

com.onlineinsight.conjointengine.
adaptive.ACADefaultPairsQuestion
ChooserImpl.
pairsattributesettingsxml

Description This property is used with the default method for pair question generation. It specifies the name of the XML file that contains attribute settings for question generation.

Valid Values The absolute file name of the attribute-setting XML file used during pairs generation, if the default way of of generating pairs questions is used.

Example com.onlineinsight.conjointengine.adaptive.
ACADefaultPairsQuestionChooserImpl.
pairsattributesettingsxml =
c:/pc/conf/pairsattributesettings.xml

com.onlineinsight.conjointengine.
adaptive.ACAFlexNumberPairs
QuestionChooserImpl.
attrNumberSettings

Description This property is used if the flex method of generating pairs questions is used. It specifies the number of attributes to use for each pair question. This is a comma-separated list of the number of attributes to use for each question. That is, the first number specifies how many attributes to use in the first question, the second

number specifies the number of attributes to use in the second question, and so on. There should be a list of 7 numbers since there are a total of 7 pairs questions.

Valid Values A list of numbers that range from 2 to N, where N is the number of attributes in the study. There need to be 7 numbers in the list, 1 for each pair question to be asked.

Example com.onlineinsight.conjointengine.adaptive.
ACAFlexNumberPairsQuestionChooserImpl.
attrNumberSettings = 2,3,3,3,3,3,4

com.onlineinsight.conjointengine.
calibrationquestionchooserimplclass

Description This property specifies the name of the class used to generate calibration questions.

Valid Values com.onlineinsight.conjointengine.adaptive.
ACABestWorstCalibrationQuestionChooserImpl
or a fully qualified name of a class that implements the com.onlineinsight.intercomponent.conjointengine.
CalibrationQuestionChooser interface.

Example com.onlineinsight.conjointengine.
calibrationquestionchooserimplclass =
com.onlineinsight.conjointengine.adaptive.
ACABestWorstCalibrationQuestionChooserImpl

com.onlineinsight.conjointengine.
adaptive.ACACalibrationQuestion
ChooserBaseImpl.numberofproducts

Description This property sets the number of calibration questions asked per interview.

Valid Values A number greater than 0.

Example com.onlineinsight.conjointengine.adaptive.
ACACalibrationQuestionChooserBaseImpl.numberofproducts = 4

Datastore Properties

The Datastore depends on an external JDBC-compliant database to operate. Therefore, many of the values of properties in this file depend on the database. For example, the user_name property must be set to the value that the database depends on, and the valid value cannot be specified in this document

server_name

Description The Datastore's name. This name is used to register with the RMI registry. For the CommandProcessor that uses the data, this name should be the same as the DatastoreName in the CommandProcessor properties file.

Valid Values The RMI name of the server. This can be any value that RMI accepts.

Example server_name= PC2

host

Description The name or IP address of the host that the Datastore is running.

Example host = 10.1.1.100

port

Description The port number that the Datastore component uses.

Valid Values A valid port number. Ports between 0 and 1023 are reserved as Well-Known Port Numbers: these ports are reserved and should not be used by any Precision Choice component. See RFC 1700 for more details on Well-Known Port Numbers.

Example port = 1299

max_database_connections

Description The maximum number of database connections to use at one time.

Valid Values A number between 0 and N, where N is the maximum number of connections that the database vendor supports.

Example max_database_connections=5

max_threads

Description The storage of interview results is asynchronous, i.e., the Datastore saves the interview results without having the client of the Datastore wait for the save to occur. To save the interview results asynchronously, the Datastore creates a pool of threads to save the results. max_threads sets the number of threads in the thread pool. This maximum can be set to any number greater than 0. Note that there is overhead associated with creating and using threads so take care when setting this value. Setting this value very high does not necessarily mean that the Datastore saves results faster.

Valid Values A number greater than 0.

Example max_threads=3

errorlog

Description This property contains the name of the file used to log errors.

Valid Values Any filename (does not need to be an absolute file name). If this is not set to an absolute file name, the log file is created in the directory from which the Datastore is started.

Example errorlog=datastore.err

database_name

Description The Datastore connects to a database to store and retrieve information. This property sets the name of the database. The Datastore passes this value directly to the database via the database driver specified in database_driver property.

Valid Values The valid value for this property is vendor-specific, i.e., this property needs to be set to a value that the vendor database can use.

Example database_name=PC

user_name

Description This is the user name, as specified by the vendor database. See the database vendor's documentation for instructions on how to set this value. The Datastore passes this value directly to the database via the database driver specified in database_driver property.

Valid Values The valid values for this property depend on the database vendor.user_name

Example user_name=pc

password

Description This is the database vendor's password for the user whose username is specified by user_name. The Datastore passes this value directly to the database via the database driver specified in database_driver property.

Valid Values The valid values for this property depend on the database vendor.

Example password=pc

database_url

Description This property sets the URL for the database. The Datastore passes this value directly to the database via the database driver specified in database_driver property.

Valid Values The valid values for this property depend on the database vendor.

Example database_url=jdbc:oracle:oci8:@pc1

database_driver

Description This property sets the database driver the Datastore uses to connect to a third-party database. The Datastore uses JDBC to communicate with the database, so this must be the fully qualified class name of the vendor's JDBC driver. The driver class specified in this property must be in the classpath of the machine running the Datastore.

Valid Values A fully qualified JDBC driver class name.

Example database_driver=oracle.jdbc.driver.OracleDriver

Interview XML File

The Interview XML file contains an XML document that describes how the Interviewer conducts an interview. The file actually describes the sections contained within the interview. Each section represents a type of question asked during the interview.

DTD:

<!ELEMENT interview (section+) >

<!ELEMENT section EMPTY >

<!ATTLIST section class NMTOKEN #REQUIRED >

interview

Description This element describes which sections are in the interview and is the root node for this document. This element contains all sections to be used in the interview.

Example of interview Element

```
<?xml version="1.0" encoding="UTF-8"?>
<interview>
<section class="com.onlineinsight.interviewer.
sections.conjoint.ratings.Ratings">
</section>
<section class="com.onlineinsight.interviewer.
sections.conjoint.importance.Importance">
</section>
<section class="com.onlineinsight.interviewer.
sections.conjoint.pairs.Pairs">
</section>
<section class="com.onlineinsight.interviewer.
```

sections.conjoint.calibration.Calibration">
</section>
</interview>

section

Description This element of the XML document describes a section of the interview. The class attribute contains the fully qualified name of the class that represents the section. Each section represents a type of question to ask during the interview. Typically, each section generates a set of questions that the Interviewer component asks. For example, if a pairs section is present in the interview, the class named in that section generates several pair questions. Sections need not generate conjoint questions. They can be used to ask any type of question. The only requirement for a section is that the class that represents it needs to extend com.onlineinsight.interviewer.Section.

There is one predefined section subtype. It is the conjoint section, and it extends com.onlineinsight.interviewer.Section. To create a new Conjoint section, the new section would need to extend com.onlineinsight.interviewer.conjoint.ConjointSection.

Valid Values com.onlineinsight.interviewer.sections.conjoint.
ratings.Ratings
com.onlineinsight.interviewer.sections.conjoint.
importance.Importance
com.onlineinsight.interviewer.sections.
conjoint.pairs.Pairs
com.onlineinsight.interviewer.sections.conjoint.
calibration.Calibration
or any class that extends
com.onlineinsight.interviewer.Section

Example <section class="com.onlineinsight.interviewer.
sections.conjoint.calibration.Calibration">
</section>

Units XML File

This file is specified in precisionchoice.properties file for specifying the display format for values of attributes in the study. These units are only used for formatting types of number or currency. Text is handled as a default type or unit.

DTD:

```
<!-- This xml document defines the units -->
<!-- for Precision Choice and how to display -->
<!-- values associated with a unit. -->
<!DOCTYPE DOCUMENT [
<!-- Root node for document -->
<!ELEMENT units ( unit+ ) >
<!-- Defines a unit for Precision Choice -->
<!ELEMENT unit ( display ) >
<!-- Name of unit -->
<!-- Examples: currency, inches, pounds -->
<!ATTLIST unit name NMTOKEN #REQUIRED >
<!-- Defines how to display a value of this unit -->
<!ELEMENT display EMPTY >
<!-- Can be either "currency" or "format"  -->
<!ATTLIST display format NMTOKEN #REQUIRED >
<!-- See java.text.DecimalFormat for description of format -->
<!-- characters. Example: "#.# lbs" -->
<!ATTLIST display pattern CDATA #REQUIRED >
]>
```

units

Description This is the root node for the document. It contains all unit descriptions.

Example of units Element

```
<?xml version="1.0" encoding="UTF-8"?>
```

<units>
<!-- formats: currency, number -->
<!-- pattern: see java.text.DecimalFormat -->
<unit name="currency">
<display format="currency"></display>
</unit>
<unit name="inches">
<display format="number" pattern="#.# in"></display>
</unit>
<unit name="megabytes">
<display format="number" pattern="#.# MB"></display>
</unit>
<unit name="gigabytes">
<display format="number" pattern="#.# GB"></display>
</unit>
<unit name="pounds">
<display format="number" pattern="#.# lbs"></display>
</unit>
<unit name="megahertz">
<display format="number" pattern="#.# MHz"></display>
</unit>
</units>

unit

Description The name by which the Interviewer component references the unit.

Example <unit name="megahertz">
<display format="number" pattern="#.# MHz"></display>
</unit>

display

Description The display element is used to format the value of an level (in an attribute) to a string. The format is set to either to number or currency. The pattern specifies how to display the particular unit, using

java.text.NumberFormatter to format the value to text.

Example <display format="number" pattern="#.# MHz"></display>

Pair Attribute Settings XML File

This XML file, identified in the precisionchoice.properties file, specifies the default attribute pairs presented to the client, if default attribute pairs is chosen as the question chooser (in the precisionchoice.properties file). The specification for default pairs depends on the total number of attributes included in the pair questions and the number of pair questions asked. The default method of pair question generation uses this file.

The following example shows how this file is used. Consider this segment of an actual Pair Attributes Setting file:

<attributesetting totalattributes="2">
<pairs totalpairs="1">
<pair order="1">
<attribute order="1">
</attribute>
<attribute order="2">
</attribute>
</pair>
</pairs>

In this example, the pair questions only use two attributes and only one pair question is asked. When the first (and only) pair question is generated, the engine looks in this file for an attributesetting whose

totalattributes equals 2. Then, it looks in that attributesetting for a pairs element whose totalpairs is equal to 1, and then it looks for a pair element whose order is 1.

When the engine generates pair questions, it looks for an attributesetting equal to the number of attributes to be used in the pair questions. Within that attributesetting

it looks for a pairs element whose totalpairs is equal to the number of pair questions to be asked. Finally, it looks within the pairs element, for a pair element whose order is equal to the current pair question. In this case, it looks for the pair element whose order is equal to 1 because there is only one pair question to be asked, and the current pair question is the first question. Within the pair element, the attributes to be used are listed by order of the attributes in the study. In this example, the pair question that is generated contains levels from the first and second attributes in the study.

Fortunately, the version of this file that comes with the Precision Choice engine is complete: it contains information for generating pair questions for up to five attributes, the maximum currently allowed for pair questions.

DTD:

Document Type Declaration for pairattributesettings.xml is as follows:

```
<!DOCTYPE DOCUMENT [
<!ELEMENT attributesettings ( attributesetting+ ) >
<!ELEMENT attributesetting ( pairs+ ) >
<!ATTLIST attributesetting totalattributes CDATA #REQUIRED >
<!ELEMENT pairs ( pair+ ) >
<!ATTLIST pairs totalpairs CDATA #REQUIRED >
<!ELEMENT pair ( attribute+ ) >
<!ATTLIST pair order CDATA #REQUIRED >
<!ELEMENT attribute EMPTY >
<!ATTLIST attribute order CDATA #REQUIRED >
]>
```

attributesettings

Description The root node for the XML document.

attributesettting

Description There is one of these elements for each possible number of attributes to be used in pair questions. For instance, the maximum number of attributes that can be used in pair questions is 5. So the file contains attributesetting whose totalattributes equals 2, 3, 4, and 5.

pairs

Description This element refers to the total number of pairs to be used in an interview. The totalpairs is set to the number of pair questions to be asked in a particular interview. For example, if 5 pair questions are to be asked, then the engine uses the pairs element (within the proper attributesetting element) whose totalpairs is equal to 5.

pair

Description For each pair question that can be asked, there is a pair section. The order refers to the order of the question in the interview. For the third question in an interview, the engine uses the pair element (within the proper pairs element) whose order is equal to 3.

attribute

Description This is a study attribute used in the pair questions. The order refers to the index of the attribute in the list of attributes to be used in the pair questions. For example, an attribute whose order is 4 is the fourth attribute in the list of available study attributes.

Understanding and Using Precision Choice Preference Profiles

Overview

Precision Choice has the ability to produce a precise profile for each end-user who participates in an interview. This preference profile is a concise capsule of information that describes how that individual end-user makes a purchasing decision. The profile can be used in many ways, one of which is to rank products from a product database.

Scores and Utilities

While the Precision Choice engine conducts an interview, it collects and calculates utilities for each attribute and level in the study. A utility is a score for the level, and this score reflects the end-user's preference for that level. Most utilities are intermediate values that are used to calculate the final set of utilities. The ultimate goal

of the engine is to produce two sets of scores: Final Scaled Utilities and Relative Importance. These scores make up Precision Choice's preference profile. The profile is retrieved from the engine using the GETSCORES command.

Final Scaled Utilities

The Final Scaled Utilities are the most important scores the engine produces because they express the end-user's preferences for levels in the study. These numerical scores indicate how appealing each level of the study is to the user. The higher the score, the higher that level's appeal is to the end-user.

As an example, consider the Final Scaled Utilities for the example study used throughout this manual. Precision Choice's engine calculated these scores as the end-user responded to the questions that Precision Choice asked.

Attribute	Level	Final Scaled Utility
Category	Aggressive Growth	60
	Growth	26
	Small Company	18
	International Stock	22
	Balanced	12
	Equity Income	18
	Corporate Bond	19
	Municipal Bond	0
Morningstar Rating	5	37
	4	33
	3	22
	2	18
	1	0
Morningstar Risk	0.5	40
	0.75	24
	1.0	15
	1.25	9

	1.5	0
Three-Year Return	30	28
	20	25
	10	19
	0	0
Load	0	32
	2	23
	4	17
	6	0

For each level, the Final Scaled Utility indicates how desirable the end-user finds that level. For example, a Load of "2" has a "utility score" of 23." A load of "0" is more appealing, having a "utility score" of 32. A Final Scaled Utility is a "utility score" for a level.

Note that the Final Scaled Utility for each level can be compared to the other levels. This allows for analysis within each attribute and across the study. For an interesting example of analysis within an attribute, look at the Category attribute in the Table above. Based on these scores, the end-user prefers Aggressive Growth first, followed by Growth, and then International Stock. Also, note that the scores can be analyzed for the study as a whole. For instance, the Aggressive Growth level in the attribute Category scored higher than any other level in the study; this end-user finds Aggressive Growth mutual funds most appealing. In fact, this user would prefer to buy an Aggressive Growth fund over a fund with a Morningstar Rating of 5. These scores are additive, i.e., you can calculate a total utility score for a product by summing the utility for each level in the product.

Fund	Category	Mstar Rating	Mstar Risk	3-Year Return	Load
Big Bank Fund	Aggressive Growth	5	0.5	20	2
Big	Corporate	2	0.5	10	0

Company Bond	Bond				

Both products are hypothetical and are used to demonstrate how final scaled utilities are additive. To score each product, the scores for each level are summed. Using the Final Scaled Utilities above, the Big Bank Fund would score 185:

60 for "Aggressive Growth"

37 for "Morningstar Rating of 5"

40 for "Morningstar Risk of 0.5"

25 for "Three-Year Return of 20"

23 for "Load of 2"

Big Company Bond would score 128:

19 for "Corporate Bond"

18 for "Morningstar Rating of 2"

40 for Morningstar Risk of 0.5"

19 for Three-Year Return of 10"

32 for "Load of 0"

This end-user would find the Big Bank Fund more appealing than the Big Company Bond. The complete method for scoring actual products is discussed below.

Relative Importance

Another use of Final Scaled Utilities is to calculate Relative Importance. Relative Importance measures the relative weight every attribute in the study has on the user's purchasing decision. This score is based on an analysis of the spread of final scaled utilities within an attribute, i.e., an analysis of the difference between the highest final scaled utility and the lowest final scaled utility for each attribute. The relative importance for each attribute is expressed as a decimal value from 0 to 1, inclusive, that reflects how much each attribute contributes to the end-user's buying decision.

For an example of how Precision Choice calculates Relative Importance scores, consider the table below. It is based on the Final Scaled Utilities example presented herein.

Attribute	Highest Level	Lowest Level	Spread
Category	60 (Aggressive Growth)	0 (Municipal Bond)	60
Morningstar Rating	37 (5)	0 (1)	37
Morningstar Risk	40 (0.5)	0 (1.5)	40
Three-Year Return	28 (30)	0 (0)	28
Load	32 (0)	0 (6)	32

To calculate the Relative Importance of each attribute, sum all spreads and then divide the spread for each attribute by the sum. In this example, the sum of the spreads is 197 (60+37+40+28+32). To calculate the relative importance for each attribute, divide the spread by the sum. The table below presents the Relative Importance scores for the example used herein.

Attribute	Spread	Relative Importance Calculation	Relative Importance (Expressed as a Percentage)
Category	60	60/197 = 0.304569	30.5%
Morningstar Rating	37	37/197 = 0.152284	15.2%
Morningstar Risk	40	40/197 = 0.203046	20.3%
Three-Year Return	28	28/197 = 0.142132	14.2%
Load	32	32/197 = 0.162437	16.2%

In this example, the Category attribute contributed most to the end-user's preference, Morningstar Risk was next, and so on.

Product Scoring

The Final Scaled Utilities are used to score products in the Precision Choice engine. After the scores are retrieved, the engine can score the products in the database and return a set of product recommendations via the GETRECOMMENDATIONS

command. The Precision Choice engine uses a scoring algorithm to score each product and to rank the products. If a client wants another algorithm, Online Insight would need to develop a custom algorithm based on the information returned from the GETSCORES command.

Precision Choice Scoring Algorithm

The default scoring algorithm is called Best Match Scoring and is based on the overall fit of a product to the end-user's final scaled utilities. In this algorithm, the Precision Choice engine sums the final scaled utility values for each level that makes up the product, and this sum is the score for that product. After all products are scored, the products receive a ranking based on the percentage of the best score.

For example, if the best product has an overall utility score of 185 and the second best has a score of 128, the best product would receive a rating of 100%, and the second would receive a rating of 69%, since 185/185 is 100% and 128/185 is 69%. The total fit of the best product is 100%. The engine also calculates the fit of each level in the product for the end-user. The fit for each level indicates how well that level meets the end-user's wishes. This is a percentage of the best final scaled utility. For example, if the product level for Three-Year Return is 10, the fit for this product level would be 68% since the score for 10 in three-year return is 19, and the best score for three-year return is 28 (19/28=0.678 or 68%).

Scoring Interpolated Attributes

Before Precision Choice's scoring algorithm can be fully examined, you need to understand how the system handles interpolated values. In the study, attributes can be marked as interpolated, i.e., the levels presented in the study for an interpolated attributes are manufactured so that the levels are equidistant from each other. This produces a range of values, and the product levels fall within this range. For example, in the Mutual Fund example (presented throughout this manual) the Load attribute is interpolated and has the levels of 0, 2, 4, and 6. In the product database, the actual levels for Load fall within the range of 0 to 6 and may not equal any of the levels in the study. Some actual levels for Load are 0.0, 4.5, 5.5 and so on. As mentioned earlier, the final scaled utilities for each product level are summed together to score the product. The problem with interpolated values is that the actual product level may not have been

included in the study, so the actual level may not have a score. To handle this situation, Precision Choice uses an algorithm to calculate a score for interpolated values.

The score for a product level that has an interpolated attribute in the study is proportional to the final scaled utilities of the levels in the study that it falls between. If the product level is greater than the highest study level, the score is the same as the highest study level. If the product level is less than the lowest study level, the product level is the same score as the lowest study level. For example, look at the product called "Fund C" in the Table below. It has a Three-Year Return of 24.3. Since no level in the Three-Year Return attribute equals 24.3, its score must be calculated so that it is proportional to the two levels that it is between (20 and 30). To calculate the score, perform the following steps:

1. Find the absolute difference between the product level and the lowest scored level that the product level is between. In this case, that would be 24.3 - 20.0 = 4.3.
2. Find the difference of the final scaled utility scores of the two study levels, in this case 28 - 25 = 3.
3. Multiply these two differences, which would result in 12.9 (4.3 * 3).
4. Divide this number by the difference in the two study levels (20 - 10 = 10). This results in 1.29 or 1 after truncating to the nearest whole number.
5. Add 1 (the value from Step 4) to the lowest of the two final scaled utility values (25 in this case) = 26
6. The score for the Three-Year Return level in this product is 26.

Another way to express this calculation is with a mathematical formula. The Precision Choice engine uses the formula below to calculate a score for an interpolated value:

$$(|LP - L1| / |L2 - L1|) * |U2 - U1| + U1$$

- LP is the level of the product
- L1 and L2 are study levels: LP is between L1 and L2
- L1 is the lowest scored study level in the pair L1 and L2
- L2 is the highest scored study level in the pair L1 and L2
- U1 is the final scaled utility for L1

- U2 is the final scaled utility for L2
- L1 < LP < L2.

Product Scoring Example

This is a step-by-step example of product scoring. To score the product database, the engine steps through each product one at a time and calculates the score of the product by summing the final scaled utilities for each level in the product, taking into account interpolated attributes. In the table below, there are five products chosen from the sample product database. For this example, this table is the complete product database.

Fund	Category	Mstar Rating	Mstar Risk	1-Year Return	3-Year Return	Load	Min Invest
Fund A	Aggressive Growth	5	1.2	66.5	39.0	0.0	2500
Fund B	Aggressive Growth	5	1.4	47.4	28.1	5.0	0
Fund C	Balanced	4	0.5	28.2	24.3	0.0	2500
Fund D	Corporate Bond	4	0.9	11.4	10.0	3.8	2000
Fund E	Municipal Bond	4	1.1	8.3	0.3	4.5	1000

To score the products, the engine goes through each product, one at a time, and calculates the total of the final scaled utilities or the Total Utility for the product. To demonstrate the algorithm, the example below scores the first product. Note that product levels which were not a part of the study (Minimum Investment and 1-Year Return) receive a score of 0.

Features	Value	Utility	Fit
Category	Aggressive Growth	60	100% 60/60 = 100%
Morningstar Rating	5	37	100% 37/37 = 100%
Morningstar Risk	1.2	10	25% 10/40 = 25%
1-Year Return	66.5	0	N/A
3-Year Return	39.0	28	100% 28/28 = 100%
Load	0	32	100% 32/32 = 100%
Minimum Investment	2500	0	N/A
Total Product Utility		167	

For the first product, the total utility is 167. The utility scores for Morningstar Risk, Three-Year Return, and Load are calculated using the algorithm listed in the Scoring Interpolated Attributes section above. Note that the engine calculated the fit per level or feature of the product. The following table presents the scores for all products in this example.

Fund	Total Utility
Fund A	167 (60+37+10+0+28+32+0)
Fund B	137 (60+37+4+0+27+9+0)
Fund C	143 (12+33+40+0+26+32+0)
Fund D	108 (19+33+19+0+19+18+0)
Fund E	60 (0+33+13+0+1+13+0)

Once the engine has scored all products, it calculates the rating for each product. The rating is calculated by dividing the total utility for each product by the total utility for the best product. The Table below demonstrates that calculation. The best product in this database is Fund A with a total utility of 167, so 167 is used to calculate all the products' ratings.

Fund	Rating
Fund A	100% (167/167 = 100%)
Fund B	82% (137/167 = 82%)
Fund C	86% (143/167 = 86%)
Fund D	65% (108/167 = 65%)
Fund E	36% (60/167 = 36%)

To conclude this example, the XML is presented for partial responses to the GETSCORES and GETRECOMMENDATIONS commands for this example. First, a part of the response to a GETSCORES command is presented. In this example, only the scores for the attribute Category are present. Normally, all attributes and their levels are returned. Each study attribute has an ATTRIBUTE tag in the XML that contains the attribute name and its relative importance. Within each ATTRIBUTE, there are the levels that make up that attribute. Each LEVEL contains its Final Scaled Utility.

```
<SCORES>
<ATTRIBUTE
NAME="Category"
RELATIVEIMPORTANCE=".304569">
<LEVEL
ID="0"
NAME="Aggressive Growth"
VALUE="Aggressive Growth">
<UTILITY NAME="FinalScaledUtilities" VALUE="60" />
</LEVEL>
<LEVEL
```

```
ID="1"
NAME="Growth"
VALUE="Growth">
<UTILITY NAME="FinalScaledUtilities" VALUE="26" />
</LEVEL>
<LEVEL
ID="2"
NAME="Small Company"
VALUE="Small Company">
<UTILITY NAME="FinalScaledUtilities" VALUE="18" />
</LEVEL>
<LEVEL
ID="3"
NAME="International Stock"
VALUE="International Stock">
<UTILITY NAME="FinalScaledUtilities" VALUE="22" />
</LEVEL>
<LEVEL
ID="4"
NAME="Balanced"
VALUE="Balanced">
<UTILITY NAME="FinalScaledUtilities" VALUE="12" />
</LEVEL>
<LEVEL
ID="5"
NAME="Equity Income"
VALUE="Equity Income">
<UTILITY NAME="FinalScaledUtilities" VALUE="18" />
</LEVEL>
<LEVEL
ID="6"
```

NAME="Corporate Bond"
VALUE="Corporate Bond">
<UTILITY NAME="FinalScaledUtilities" VALUE="19" />
</LEVEL>
<LEVEL
ID="7"
NAME=" Municipal Bond"
VALUE=" Municipal Bond">
<UTILITY NAME="FinalScaledUtilities" VALUE="0" />
</LEVEL>
</ATTRIBUTE>
</SCORES>

The XML for a response to the GETRECOMMENDATIONS command is shown below. The MAXPRODUCTS tag refers to the total number of products in the database. This is used with RANK so that the end-user can tell how far into the ranked products he or she is. For example, a presentation layer could display that the end-user is viewing product 1 of 5. The levels of the product and which attribute the level is associated with are identified inside each PRODUCT tag. Each product has a fit in this XML, which is the rating score calculated earlier. Each product level is presented in an ATTRIBUTE tag, and the VALUE refers to the level's name. In the first product for the ATTRIBUTE Category, the product level is Aggressive Growth.

Additionally, two scores are sent with each recommendation. The first is the fit of the level to the user's need. In the first product, the attributes Category, Morningstar Rating, Three Year Return, and Load meet 100% of the end-user's needs. The attribute Morningstar Risk only met 25% of the user's needs. The fit can be used with the RELATIVEIMPORTANCE score to determine that the first product meets 100% of the user's needs in this category, which is the most important attribute. Attributes not included in the study receive a fit of -1. The GETRECOMENDATIONS command response provides information to display to the end-user to inform him or her how each product received its recommendation.

<MAXPRODUCTS VALUE="5"/>

```
<PRODUCTS>
<PRODUCT ID="1" NAME="A" FIT="100" RANK="1" >
<ATTRIBUTE
NAME="Category"
VALUE="Aggressive Growth"
FIT="100.0"
RELATIVEIMPORTANCE="30.5"/>
<ATTRIBUTE
NAME="Morningstar Rating"
VALUE="5"
FIT="100.0"
RELATIVEIMPORTANCE="15.2"/>
<ATTRIBUTE
NAME="Morningstar Risk"
VALUE="1.2"
FIT="25.0"
RELATIVEIMPORTANCE="20.3"/>
<ATTRIBUTE
NAME="One Year Return"
VALUE="66.5"
FIT="-1.0"
RELATIVEIMPORTANCE="0"/>
<ATTRIBUTE
NAME="Three Year Return"
VALUE="39.0"
FIT="100.0"
RELATIVEIMPORTANCE="14.2"/>
<ATTRIBUTE
NAME="Load"
VALUE="0.0"
FIT="100.0"
```

```
RELATIVEIMPORTANCE="16.2"/>
<ATTRIBUTE
NAME="Minimum Investment"
VALUE="2500"
FIT="-1.0"
RELATIVEIMPORTANCE="0"/>
</PRODUCT>
<PRODUCT
ID="3"
NAME="C"
FIT="86.0"
RANK="2" >
<ATTRIBUTE
NAME="Category"
VALUE="Balanced"
FIT="20.0"
RELATIVEIMPORTANCE="30.5"/>
<ATTRIBUTE
NAME="Morningstar Rating"
VALUE="4"
FIT="89.0"
RELATIVEIMPORTANCE="15.2"/>
<ATTRIBUTE
NAME="Morningstar Risk"
VALUE="0.5"
FIT="100.0"
RELATIVEIMPORTANCE="20.3"/>
<ATTRIBUTE
NAME="One Year Return"
VALUE="28.2"
FIT="-1.0"
```

```
RELATIVEIMPORTANCE="0"/>
<ATTRIBUTE
NAME="Three Year Return"
VALUE="24.3"
FIT="93.0"
RELATIVEIMPORTANCE="14.2"/>
<ATTRIBUTE
NAME="Load"
VALUE="0.0"
FIT="100.0"
RELATIVEIMPORTANCE="16.2"/>
<ATTRIBUTE
NAME="Minimum Investment"
VALUE="2500"
FIT="-1.0"
RELATIVEIMPORTANCE="0"/>
</PRODUCT>
<PRODUCT
ID="2"
NAME="B"
FIT="82.0"
RANK="3" >
<ATTRIBUTE
NAME="Category"
VALUE="Aggressive Growth"
FIT="100.0"
RELATIVEIMPORTANCE="30.5"/>
<ATTRIBUTE
NAME="Morningstar Rating"
VALUE="5"
FIT="100.0"
```

```
RELATIVEIMPORTANCE="15.2"/>
<ATTRIBUTE
NAME="Morningstar Risk"
VALUE="1.4"
FIT="10.0"
RELATIVEIMPORTANCE="20.3"/>
<ATTRIBUTE
NAME="One Year Return"
VALUE="47.4"
FIT="-1.0"
RELATIVEIMPORTANCE="0"/>
<ATTRIBUTE
NAME="Three Year Return"
VALUE="28.1"
FIT="100.0"
RELATIVEIMPORTANCE="14.2"/>
<ATTRIBUTE
NAME="Load"
VALUE="5.0"
FIT="28.0"
RELATIVEIMPORTANCE="16.2"/>
<ATTRIBUTE
NAME="Minimum Investment"
VALUE="0"
FIT="-1.0"
RELATIVEIMPORTANCE="0"/>
</PRODUCT>
<PRODUCT
ID="4"
NAME="D"
FIT="65.0"
```

```
RANK="4" >
<ATTRIBUTE
NAME="Category"
VALUE="Corporate Bond"
FIT="32.0"
RELATIVEIMPORTANCE="30.5"/>
<ATTRIBUTE
NAME="Morningstar Rating"
VALUE="4"
FIT="89.0"
RELATIVEIMPORTANCE="15.2"/>
<ATTRIBUTE
NAME="Morningstar Risk"
VALUE="0.9"
FIT="48.0"
RELATIVEIMPORTANCE="20.3"/>
<ATTRIBUTE
NAME="One Year Return"
VALUE="11.4"
FIT="-1.0"
RELATIVEIMPORTANCE="0"/>
<ATTRIBUTE
NAME="Three Year Return"
VALUE="10.0"
FIT="68.0"
RELATIVEIMPORTANCE="14.2"/>
<ATTRIBUTE
NAME="Load"
VALUE="3.8"
FIT="56.0"
RELATIVEIMPORTANCE="16.2"/>
```

<ATTRIBUTE
NAME="Minimum Investment"
VALUE="2000"
FIT="-1.0"
RELATIVEIMPORTANCE="0"/>
</PRODUCT>
<PRODUCT
ID="5"
NAME="E"
FIT="36.0"
RANK="5" >
<ATTRIBUTE
NAME="Category"
VALUE="Municipal Bond"
FIT="0.0"
RELATIVEIMPORTANCE="30.5"/>
<ATTRIBUTE
NAME="Morningstar Rating"
VALUE="4"
FIT="89.0"
RELATIVEIMPORTANCE="15.2"/>
<ATTRIBUTE
NAME="Morningstar Risk"
VALUE="1.1"
FIT="33.0"
RELATIVEIMPORTANCE="20.3"/>
<ATTRIBUTE
NAME="One Year Return"
VALUE="8.3"
FIT="-1.0"
RELATIVEIMPORTANCE="0"/>

```
<ATTRIBUTE
NAME="Three Year Return"
VALUE="0.3"
FIT="4.0"
RELATIVEIMPORTANCE="14.2"/>
<ATTRIBUTE
NAME="Load"
VALUE="4.5"
FIT="41.0"
RELATIVEIMPORTANCE="16.2"/>
<ATTRIBUTE
NAME="Minimum Investment"
VALUE="1000"
FIT="-1.0"
RELATIVEIMPORTANCE="0"/>
</PRODUCT>
</PRODUCTS>
```

Uses for Preference Data

The above discussion demonstrated how preference data can be used to score products for recommendation. There are also other ways the data can be used to provide valuable information to end-users and to vendors. This section describes other scenarios that demonstrate some uses and value of the preference data.

Notify Consumers of New Products that Meet Their Preferences

Preference data can be used to rank the appeal of new products. Imagine that an end-user completed an entire interview but did not purchase a product. Imagine further that a company introduced a new mutual fund a week later. The same preference profile could be used to determine the total utility score for the new fund. If it is above a certain threshold, the end-user could be notified that a new fund fitting his or her needs had just become available. This notification could occur in many ways, e.g., through email or a popup window on the end-user's browser the next time he or she logs in to the system.

Aggregated Preference Data

Finally, preference data can be aggregated, i.e., compiled for many consumers and then analyzed for new product development. The concept is to take the preference data for many users in a market and analyze it to determine which feature mix most meets the consumers' need and maximizes vendors' profit.

The embodiments described above are given as illustrative examples only. It will be readily appreciated that many deviations may be made from the specific embodiment disclosed in this specification without departing from the invention. Accordingly, the scope of the invention is to be determined by the claims below rather than being limited to the specifically described embodiment above.

What is claimed is:

1. A method for providing a purchaser with purchase decision support with respect to a product type, the method comprising the steps of:
   a) conducting a configurable interview individualized to the purchaser with respect to the product type;
   b) generating individually-valid, statistics-based user preferences corresponding to the product type that are specific to the purchaser in conjunction with the interview of the purchaser; and
   c) presenting the purchaser with a purchase recommendation for products within the product type based upon the generated user preferences.

2. The method of claim 1, and further comprising the step of receiving interview configuration constraints from a configuration source.

3. The method of claim 2, wherein the configuration source is the purchaser.

4. The method of claim 2, wherein the configuration source is an internal configuration file.

5. The method of claim 2, wherein the configuration source is an external application.

6. The method of claim 5, wherein the external application provides interview configuration constraints as an XML stream.

7. The method of claim 5, wherein the external application provides interview configuration constraints as a profile of information associated with the purchaser.

8. The method of claim 2, wherein the configuration source is the generated user preferences.

9. The method of claim 1, and further comprising the step of defining attributes associated with the product type and corresponding levels of the attributes of specific relevance to the purchaser.

10. The method of claim 9, wherein the step of defining attributes comprises the step of providing the purchaser with individually tailored educational material relevant to the attributes associated with the product type.

11. The method of claim 1, and further comprising the step of providing the purchaser with individually tailored educational material relevant to the product type.

12. The method of claim 1, wherein the step of generating statistics-based user preferences comprises the step of generating conjoint analysis utilities corresponding to the product type that are specific to the purchaser through the conducted interview.

13. The method of claim 12, wherein the step of generating conjoint analysis utilities comprises the step of generating priors utilities corresponding to the product type that are specific to the purchaser through the conducted interview.

14. The method of claim 13, wherein the step of generating priors utilities comprises the steps of:

i) ranking levels within attributes associated with the product type with respect to the purchaser; and

ii) calculating priors utilities.

15. The method of claim 14, wherein the step of generating priors utilities further comprises the step of rating of the attributes importance with respect to the purchaser.

16. The method of claim 15, wherein the step of generating priors utilities further comprises the step of setting attribute constraints.

17. The method of claim 14, wherein the step of generating priors utilities further comprises the step of setting attribute constraints.

18. The method of claim 13, wherein the step of generating priors utilities comprises the steps of:

i) rating levels within attributes associated with the product type with respect to the purchaser; and

ii) calculating priors utilities.

19. The method of claim 18, wherein the step of generating priors utilities further comprises the step of rating of the attributes importance with respect to the purchaser.

20. The method of claim 19, wherein the step of generating priors utilities further comprises the step of setting attribute constraints.

21. The method of claim 18, wherein the step of generating priors utilities further comprises the step of setting attribute constraints.

22. The method of claim 13, wherein the step of generating priors utilities comprises the steps of:

i) rating of the attributes importance with respect to the purchaser; and

ii) calculating priors utilities.

23. The method of claim 22, wherein the step of generating priors utilities further comprises the step of setting attribute constraints.

24. The method of claim 13, wherein the step of generating priors utilities comprises the steps of:

i) setting attribute constraints; and

ii) calculating priors utilities.

25. The method of claim 12, wherein the step of generating conjoint analysis utilities comprises the step of generating pairs utilities corresponding to the product type that are specific to the purchaser through the conducted interview.

26. The method of claim 25, wherein the step of generating pairs utilities comprises the steps of:

i) determining a number of pairs questions to ask;

ii) choosing each of the determined number of pairs questions;

iii) configuring the interview with the chosen pairs questions; and

iv) calculating pairs utilities.

27. The method of claim 26, wherein the step of choosing each of the determined number of pairs questions further comprises the step of generating prohibited pairs.

28. The method of claim 12, wherein the step of generating conjoint analysis utilities comprises the steps of:

i) generating priors utilities corresponding to the product type that are specific to the purchaser through the conducted interview;

ii) generating pairs utilities corresponding to the product type that are specific to the purchaser through the conducted interview; and

iii) generating final utilities.

29. The method of claim 28, wherein the step of generating final utilities comprises weighting the priors utilities and the pairs utilities equally.

30. The method of claim 28, wherein the step of generating final utilities comprises weighting the priors utilities and the pairs utilities optimally.

31. The method of claim 30, wherein the step of generating conjoint analysis utilities further comprises the step of calibrating the final utilities.

32. The method of claim 1, wherein the step of presenting the purchaser with a purchase recommendation for products within the product type based upon the generated user preferences comprises the step of providing the purchaser with an explanation of the purchase recommendation.

33. The method of claim 1, and further comprising the step of calibrating the generated user preferences.

34. The method of claim 1, and further comprising the step of storing the generated user preferences in a data store.

35. The method of claim 34, and further comprising the step of analyzing the generated user preferences in the data store.

36. The method of claim 35, and further comprising the step of outputting the analyzed user preferences.

37. The method of claim 35, and further comprising the step of storing the analyzed user preferences in the data store.

38. The method of claim 1, and further comprising the step of outputting the generated user preferences.

39. The method of claim 1, wherein the step of presenting the purchaser with a purchase recommendation for products within the product type based upon the

generated user preferences comprises the step formatting the purchase recommendation as an XML stream.

40. The method of claim 1, wherein the step of presenting the purchaser with a purchase recommendation for products within the product type based upon the generated user preferences comprises the step of formatting the purchase recommendation as an HTML stream.

41. A purchase decision support system providing a purchaser with purchase decision support with respect to a product with a product type, the system comprising:
    a) a data store for storing purchaser preferences and product data; and
    b) a server in communication with the data store for performing the steps of:
        i) conducting a configurable interview with the purchaser with respect to the product type;
        ii) generating statistics-based user preferences corresponding to the product type that are specific to the purchaser in conjunction with the interview of the purchaser; and
        iii) presenting the purchaser with a purchase recommendation for products within the product type based upon the conjoint analysis utilities.

42. The system of claim 41, and further comprising a display in communication with the processor for displaying the purchase recommendation.

43. The system of claim 41, wherein the server performs the further step comprising of receiving interview configuration constraints from a configuration source.

44. The system of claim 43, wherein the configuration source is the purchaser.

45. The system of claim 43, wherein the configuration source is an internal configuration file.

46. The system of claim 43, wherein the configuration source is an external application.

47. The system of claim 46, wherein the external application provides interview configuration constraints as an XML stream.

48. The system of claim 46, wherein the external application provides interview configuration constraints as a profile of information associated with the purchaser.

49. The system of claim 43, wherein the configuration source is the generated user preferences.

50. The system of claim 41, wherein the server performs the further step comprising of defining attributes associated with the product type and corresponding levels of the attributes of specific relevance to the purchaser.

51. The system of claim 50, wherein server performs the step of defining attributes by performing the step comprising of providing the purchaser with individually tailored educational material relevant to the attributes associated with the product type.

52. The system of claim 41, wherein the server performs the further step comprising of providing the purchaser with individually tailored educational material relevant to the product type.

53. The system of claim 41, wherein server performs the step of generating statistics-based user preferences by performing the step comprising of generating conjoint analysis utilities corresponding to the product type that are specific to the purchaser through the conducted interview.

54. The system of claim 53, wherein server performs the step of generating conjoint analysis utilities by performing the step comprising of generating priors utilities

corresponding to the product type that are specific to the purchaser through the conducted interview.

55. The system of claim 54, wherein server performs the step of generating priors utilities by performing the steps comprising of:

(1) ranking levels within attributes associated with the product type with respect to the purchaser; and

(2) calculating priors utilities.

56. The system of claim 55, wherein server performs the step of generating priors utilities by performing the further step comprising of rating of the attributes importance with respect to the purchaser.

57. The system of claim 56, wherein server performs the step of generating priors utilities by performing the further step comprising of the step of setting attribute constraints.

58. The system of claim 55, wherein server performs the step of generating priors utilities by performing the further step comprising of setting attribute constraints.

59. The system of claim 54, wherein server performs the step of generating priors utilities by performing the steps comprising of:

(1) rating levels within attributes associated with the product type with respect to the purchaser; and

(2) calculating priors utilities.

60. The system of claim 59, wherein server performs the step of generating priors utilities by performing the further step comprising of rating of the attributes importance with respect to the purchaser.

61. The system of claim 60, wherein server performs the step of generating priors utilities by performing the further step comprising of setting attribute constraints.

62. The system of claim 59, wherein server performs the step of generating priors utilities by performing the further step comprising of setting attribute constraints.

63. The system of claim 54, wherein server performs the step of generating priors utilities by performing the steps comprising of:

(1) rating of the attributes importance with respect to the purchaser; and

(2) calculating priors utilities.

64. The system of claim 63, wherein server performs the step of generating priors utilities by performing the further step comprising of setting attribute constraints.

65. The system of claim 54, wherein server performs the step of generating priors utilities by performing the steps comprising of:

(1) setting attribute constraints; and

(2) calculating priors utilities.

66. The system of claim 53, wherein server performs the step of generating conjoint analysis utilities by performing the step comprising of generating pairs utilities corresponding to the product type that are specific to the purchaser through the conducted interview.

67. The system of claim 66, wherein server performs the step of generating pairs utilities by performing the steps comprising of:

(1) determining a number of pairs questions to ask;

(2) choosing each of the determined number of pairs questions;

(3) configuring the interview with the chosen pairs questions; and

(4) calculating pairs utilities.

68. The system of claim 67, wherein server performs the step of choosing each of the determined number of pairs questions by performing the further step comprising of generating prohibited pairs.

69. The system of claim 53, wherein server performs the step of generating conjoint analysis utilities by performing the steps comprising of:

(1) generating priors utilities corresponding to the product type that are specific to the purchaser through the conducted interview;

(2) generating pairs utilities corresponding to the product type that are specific to the purchaser through the conducted interview; and

(3) generating final utilities.

70. The system of claim 69, wherein server performs the step of generating final utilities by performing the step comprising of weighting the priors utilities and the pairs utilities equally.

71. The system of claim 69, wherein server performs the step of generating final utilities by performing the step comprising of weighting the priors utilities and the pairs utilities optimally.

72. The system of claim 71, wherein server performs the step of generating conjoint analysis utilities by performing the further step comprising of calibrating the final utilities.

73. The system of claim 41, wherein server performs the step of presenting the purchaser with a purchase recommendation for products within the product type based upon the generated user preferences by performing the step comprising of providing the purchaser with an explanation of the purchase recommendation.

74. The system of claim 41, wherein server performs the further step comprising of calibrating the generated user preferences.

75. The system of claim 41, wherein server performs the further step comprising of storing the generated user preferences in a data store.

76. The system of claim 75, wherein server performs the further step comprising of analyzing the generated user preferences in the data store.

77. The system of claim 76, wherein server performs the further step comprising of outputting the analyzed user preferences.

78. The system of claim 76, wherein server performs the further step comprising of storing the analyzed user preferences in the data store.

79. The system of claim 41, wherein server performs the further step comprising of outputting the generated user preferences.

80. The system of claim 41, wherein server performs the step of presenting the purchaser with a purchase recommendation for products within the product type based upon the generated user preferences by performing the step comprising of formatting the purchase recommendation as an XML stream.

81. The system of claim 41, wherein server performs the step of presenting the purchaser with a purchase recommendation for products within the product type based upon the generated user preferences by performing the step comprising of formatting the purchase recommendation as an HTML stream.

82. A computer-readable digital storage device containing instructions that upon execution by a processor cause the processor to perform the steps of:

a) conducting a configurable interview individualized to the purchaser with respect to the product type;

b) generating individually-valid, statistics-based user preferences corresponding to the product type that are specific to the purchaser in conjunction with the interview of the purchaser; and

c) presenting the purchaser with a purchase recommendation for products within the product type based upon the generated user preferences.

83. The digital storage device of claim 82, and containing further instructions that, upon execution by a processor, cause the processor to perform the step of generating users preferences by performing the step comprising of generating conjoint analysis utilities corresponding to the product type that are specific to the purchaser through the conducted interview.

84. A purchase decision support system providing a purchaser with purchase decision support with respect to a product with a product type, the system comprising:

a) means for conducting a configurable interview individualized to the purchaser with respect to the product type;

b) means for generating individually-valid, statistics-based user preferences corresponding to the product type that are specific to the purchaser in conjunction with the interview of the purchaser; and

c) means for presenting the purchaser with a purchase recommendation for products within the product type based upon the generated user preferences.

85. The method of claim 84, wherein the means for generating statistics-based user preferences comprises means for generating conjoint analysis utilities corresponding to the product type that are specific to the purchaser through the conducted interview.

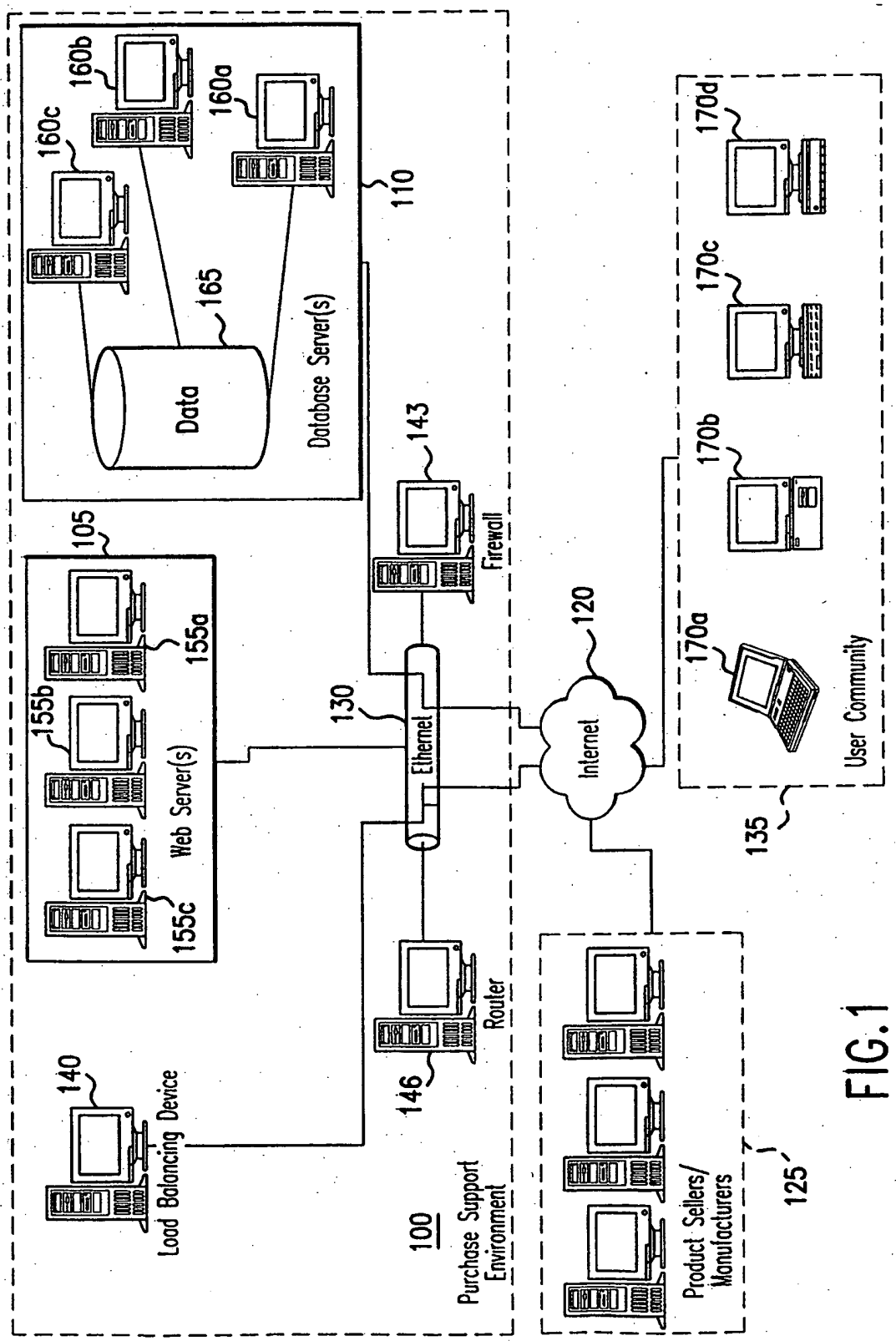

160b
160a
160c
110
165
Data
Database Server(s)
105
155a
155b
155c
Web Server(s)
143
Firewall
130
Ethernet
120
Internet
170d
170c
170b
170a
User Community
135
140
Load Balancing Device
146
Router
100
Purchase Support Environment
Product Sellers/ Manufacturers
125

FIG. 1

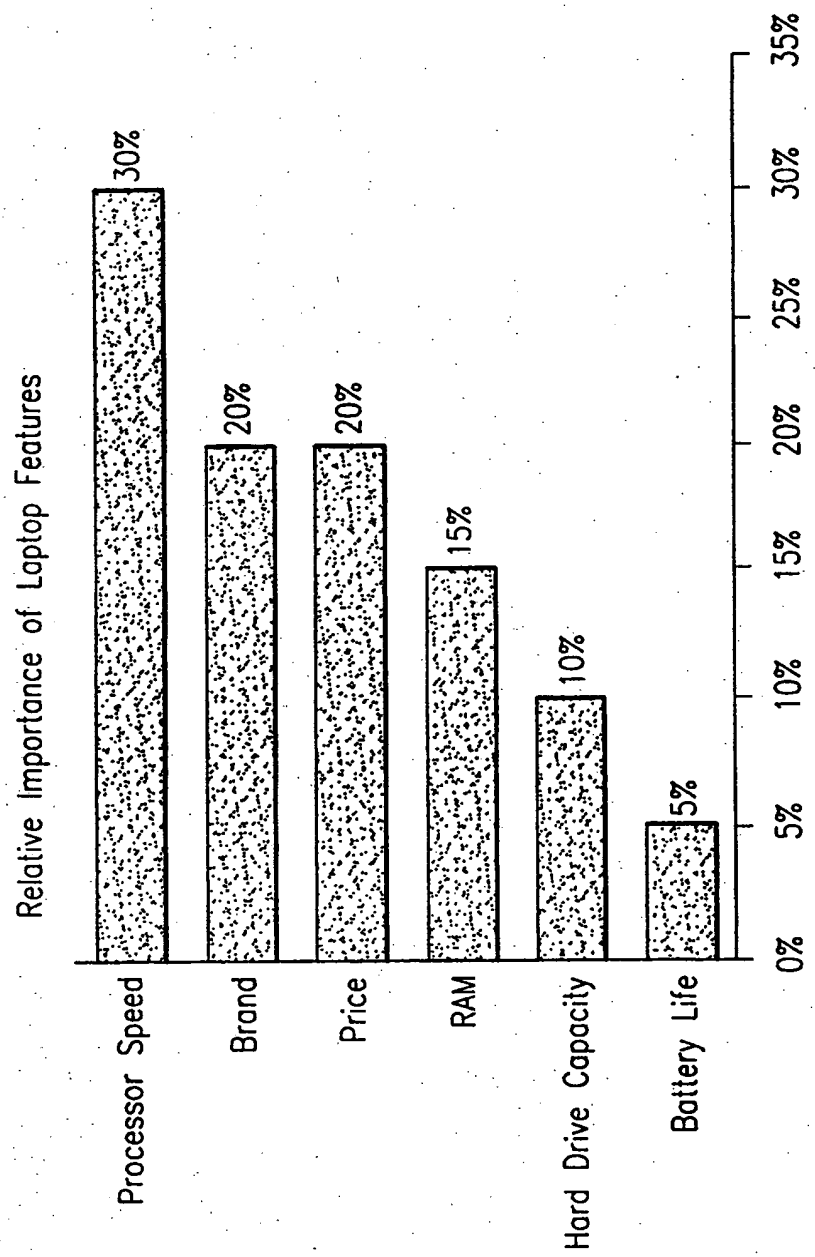

Relative Importance of Laptop Features
30%
20%
20%
15%
10%
5%
Processor Speed
Brand
Price
RAM
Hard Drive Capacity
Battery Life
0%
5%
10%
15%
20%
25%
30%
35%

FIG.2

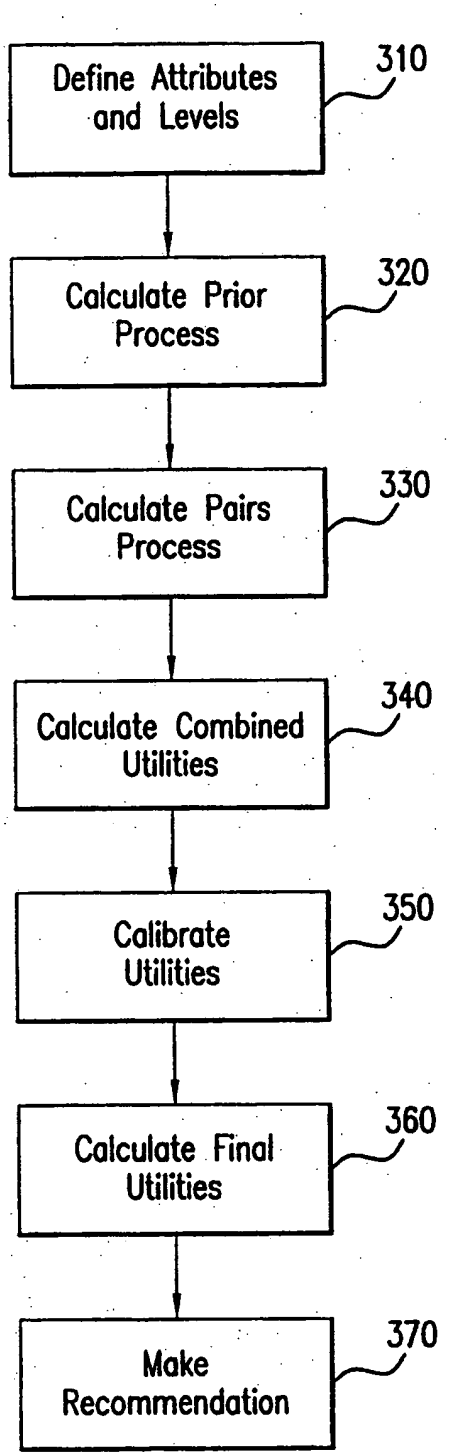

300
Define Attributes and Levels
310
Calculate Prior Process
320
Calculate Pairs Process
330
Calculate Combined Utilities
340
Calibrate Utilities
350
Calculate Final Utilities
360
Make Recommendation
370

FIG.3

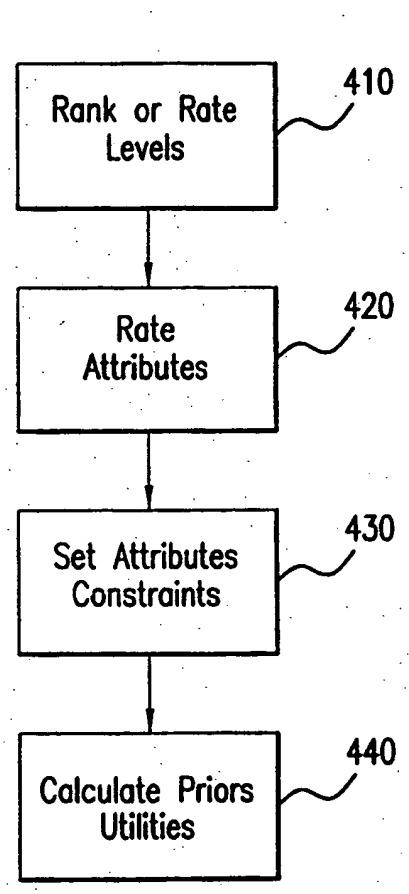

320
Rank or Rate Levels
410
Rate Attributes
420
Set Attributes Constraints
430
Calculate Priors Utilities
440

FIG.4

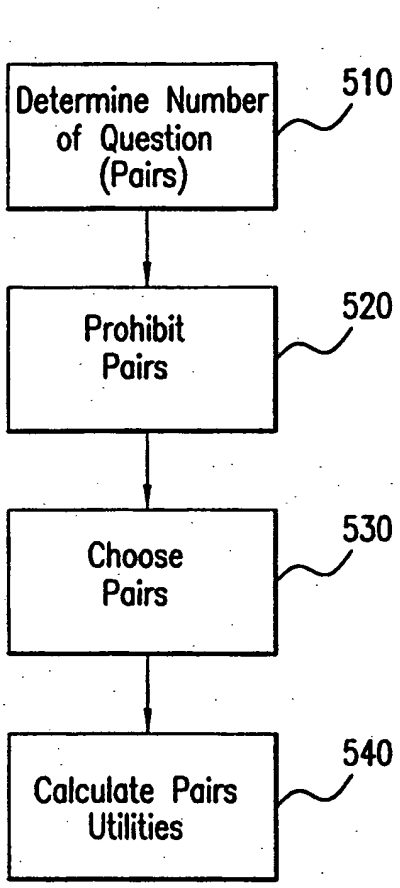

330
Determine Number of Question (Pairs)
510
Prohibit Pairs
520
Choose Pairs
530
Calculate Pairs Utilities
540

FIG.5

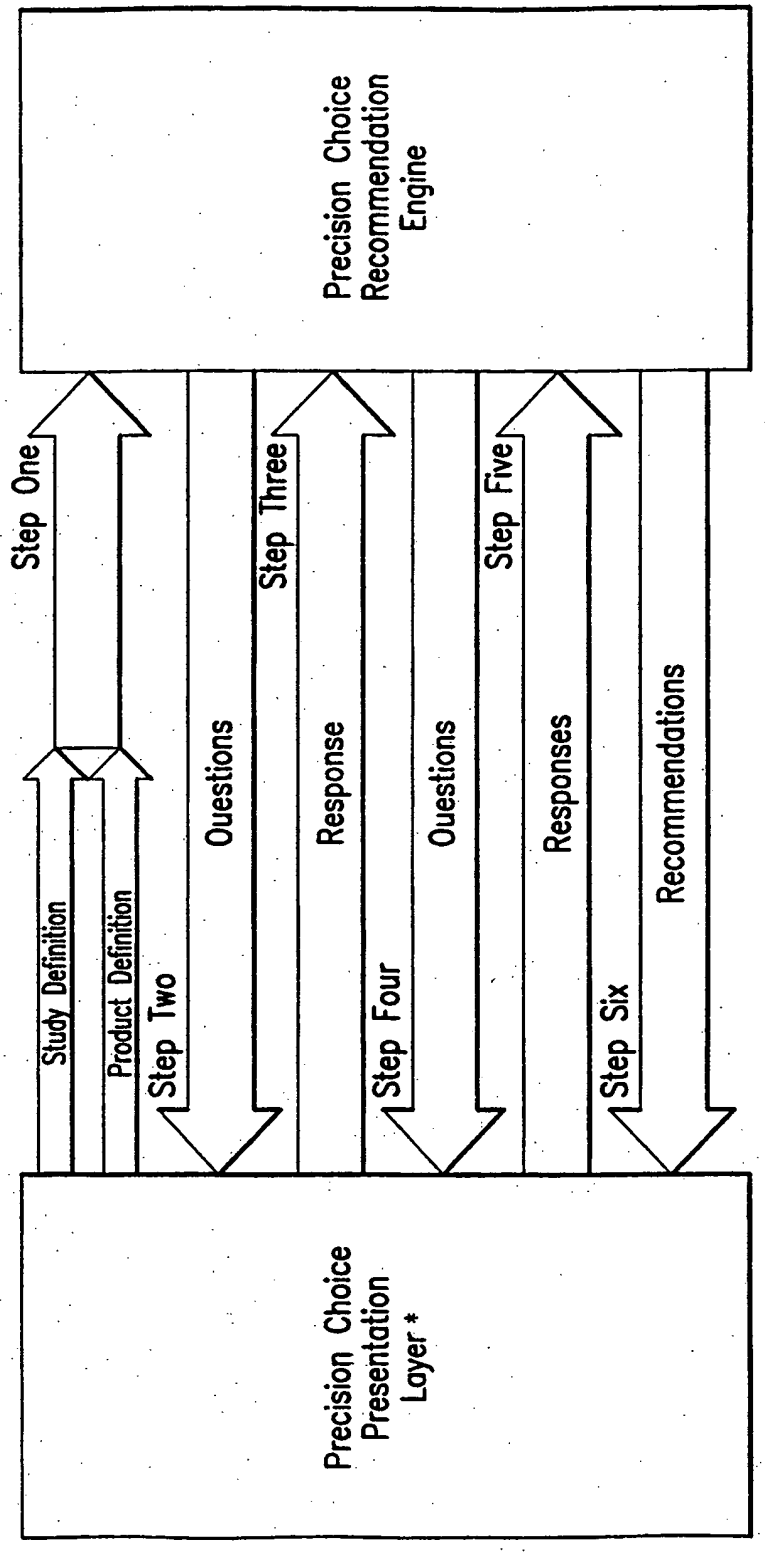

Precision Choice Recommendation Engine
Step One
Study Definition
Product Definition
Step Two
Questions
Step Three
Response
Step Four
Questions
Step Five
Responses
Step Six
Recommendations
Precision Choice Presentation Layer *

FIG.6

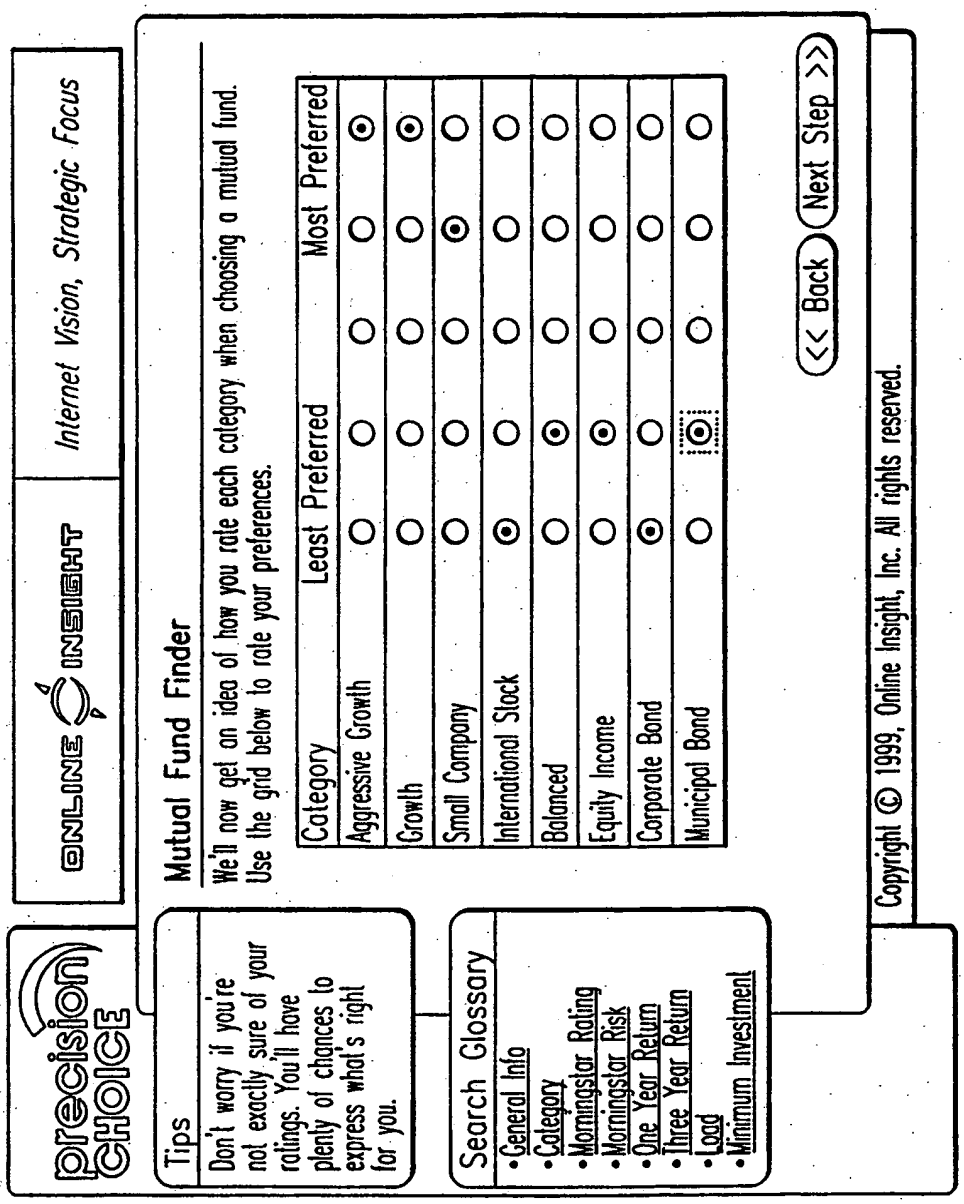

precision choice
ONLINE INSIGHT
Internet Vision, Strategic Focus
Tips
Don't worry if you're not exactly sure of your ratings. You'll have plenty of chances to express what's right for you.
Mutual Fund Finder
We'll now get an idea of how you rate each category when choosing a mutual fund. Use the grid below to rate your preferences.
Category Least Preferred Most Preferred
Aggressive Growth
Growth
Small Company
International Stock
Balanced
Equity Income
Corporate Bond
Municipal Bond
<< Back
Next Step >>
Search Glossary
General Info
Category
Morningstar Rating
Morningstar Risk
One Year Return
Three Year Return
Load
Minimum Investment
Copyright © 1999, Online Insight, Inc. All rights reserved.

FIG.7

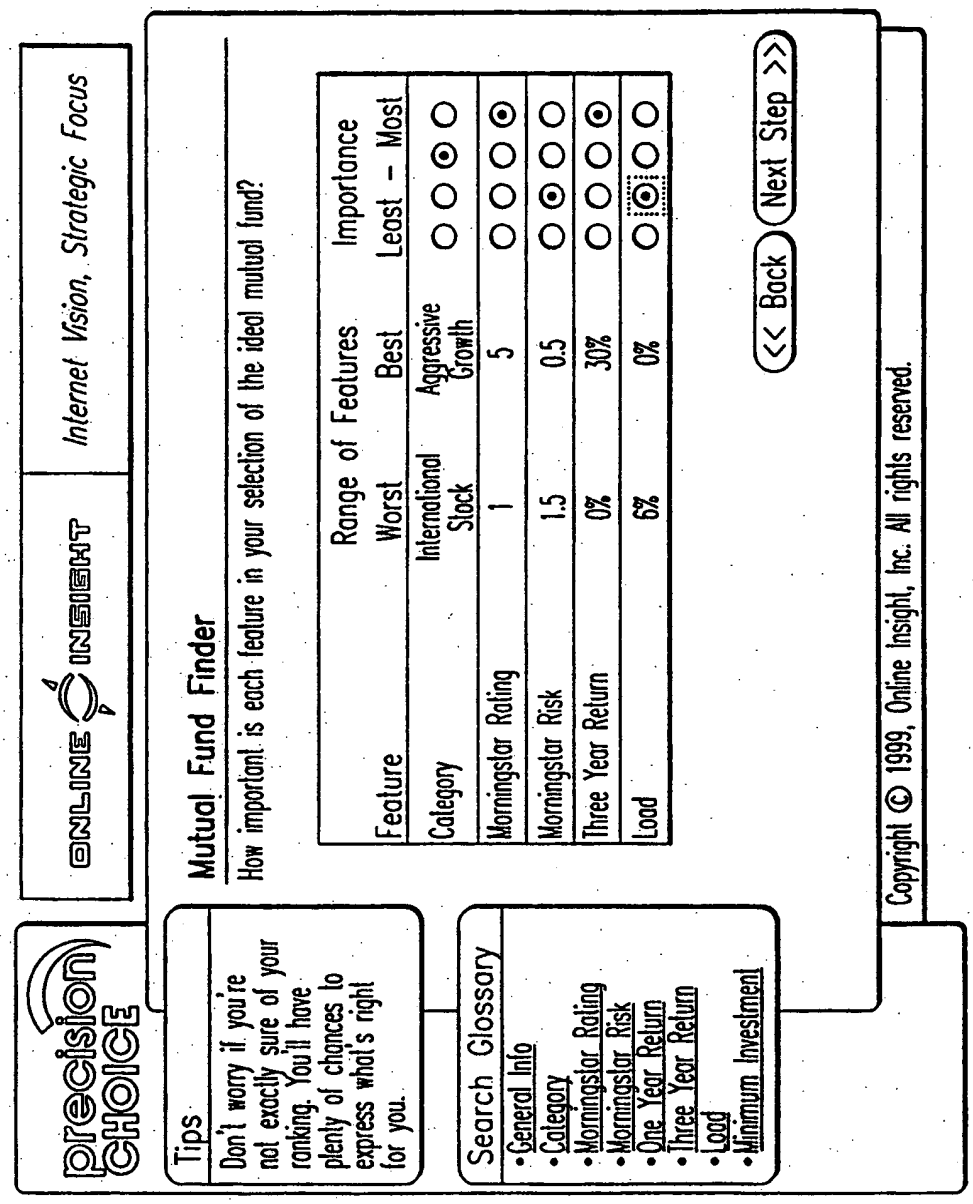

precision CHOICE
ONLINE INSIGHT
Internet Vision, Strategic Focus
Mutual Fund Finder
How important is each feature in your selection of the ideal mutual fund?
Range of Features
Feature
Worst
Best
Importance
Least - Most
Category
International Stock
Aggressive Growth
Morningstar Rating
1
5
Morningstar Risk
1.5
0.5
Three Year Return
0%
30%
Load
6%
0%
<< Back
Next Step >>
Tips
Don't worry if you're not exactly sure of your ranking. You'll have plenty of chances to express what's right for you.
Search Glossary
• General Info
• Category
• Morningstar Rating
• Morningstar Risk
• One Year Return
• Three Year Return
• Load
• Minimum Investment
Copyright © 1999, Online Insight, Inc. All rights reserved.

FIG.8

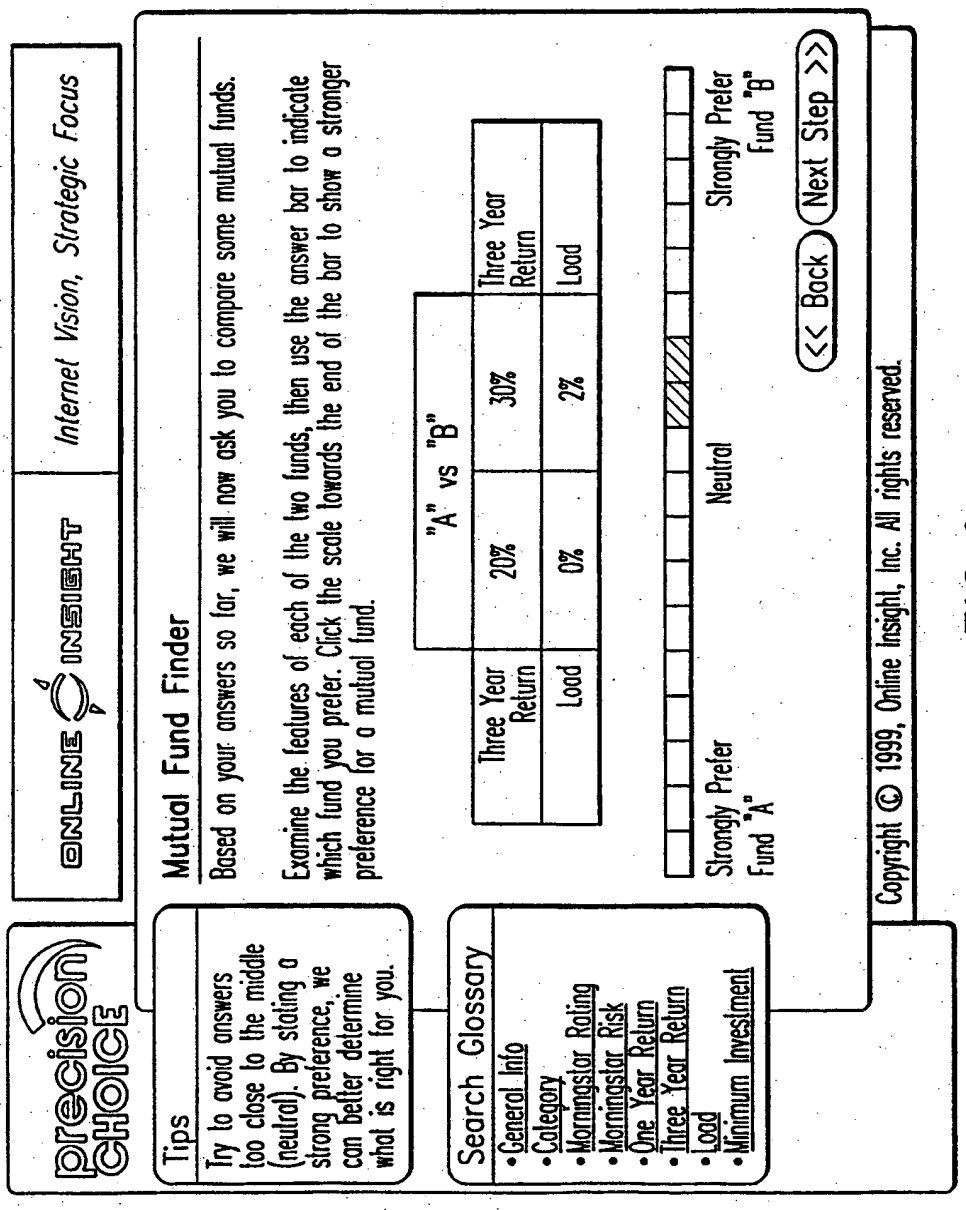

precision choice
ONLINE INSIGHT
Internet Vision, Strategic Focus
Tips
Try to avoid answers too close to the middle (neutral). By stating a strong preference, we can better determine what is right for you.
Search Glossary
• General Info
• Category
• Morningstar Rating
• Morningstar Risk
• One Year Return
• Three Year Return
• Load
• Minimum Investment
Mutual Fund Finder
Based on your answers so far, we will now ask you to compare some mutual funds.
Examine the features of each of the two funds, then use the answer bar to indicate which fund you prefer. Click the scale towards the end of the bar to show a stronger preference for a mutual fund.
"A" vs "B"
Three Year Return 20% 30% Three Year Return
Load 0% 2% Load
Strongly Prefer Fund "A"
Neutral
Strongly Prefer Fund "B"
<< Back
Next Step >>
Copyright © 1999, Online Insight, Inc. All rights reserved.

FIG.9

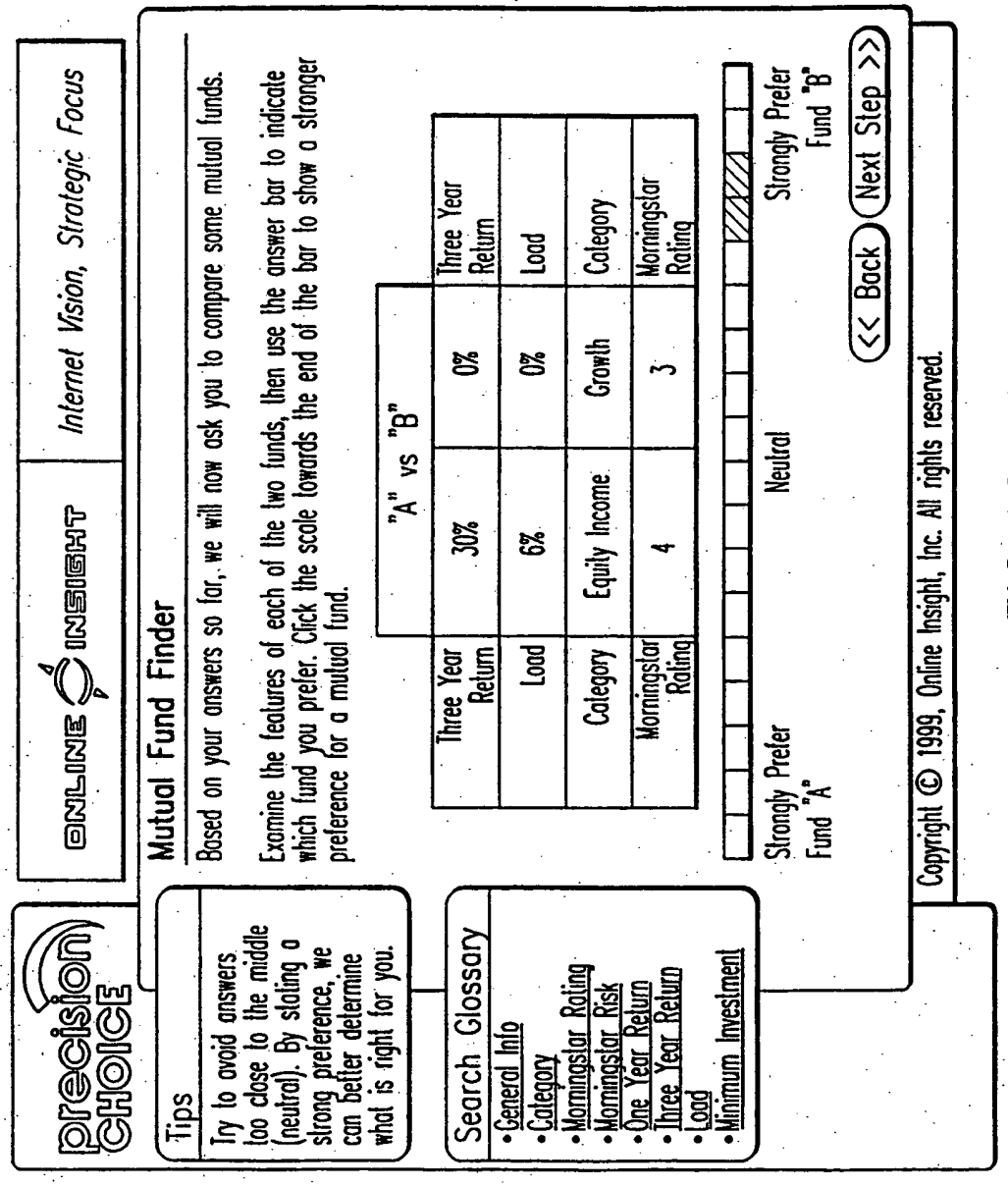

precision choice
ONLINE INSIGHT
Internet Vision, Strategic Focus
Tips
Try to avoid answers too close to the middle (neutral). By stating a strong preference, we can better determine what is right for you.
Search Glossary
•General Info
•Category
•Morningstar Rating
•Morningstar Risk
•One Year Return
•Three Year Return
•Load
•Minimum Investment
Mutual Fund Finder
Based on your answers so far, we will now ask you to compare some mutual funds.
Examine the features of each of the two funds, then use the answer bar to indicate which fund you prefer. Click the scale towards the end of the bar to show a stronger preference for a mutual fund.
"A" vs "B"
Three Year Return 30% 0% Three Year Return
Load 6% 0% Load
Category Equity Income Growth Category
Morningstar Rating 4 3 Morningstar Rating
Strongly Prefer Fund "A"
Neutral
Strongly Prefer Fund "B"
<< Back
Next Step >>
Copyright © 1999, Online Insight, Inc. All rights reserved.

FIG. 10

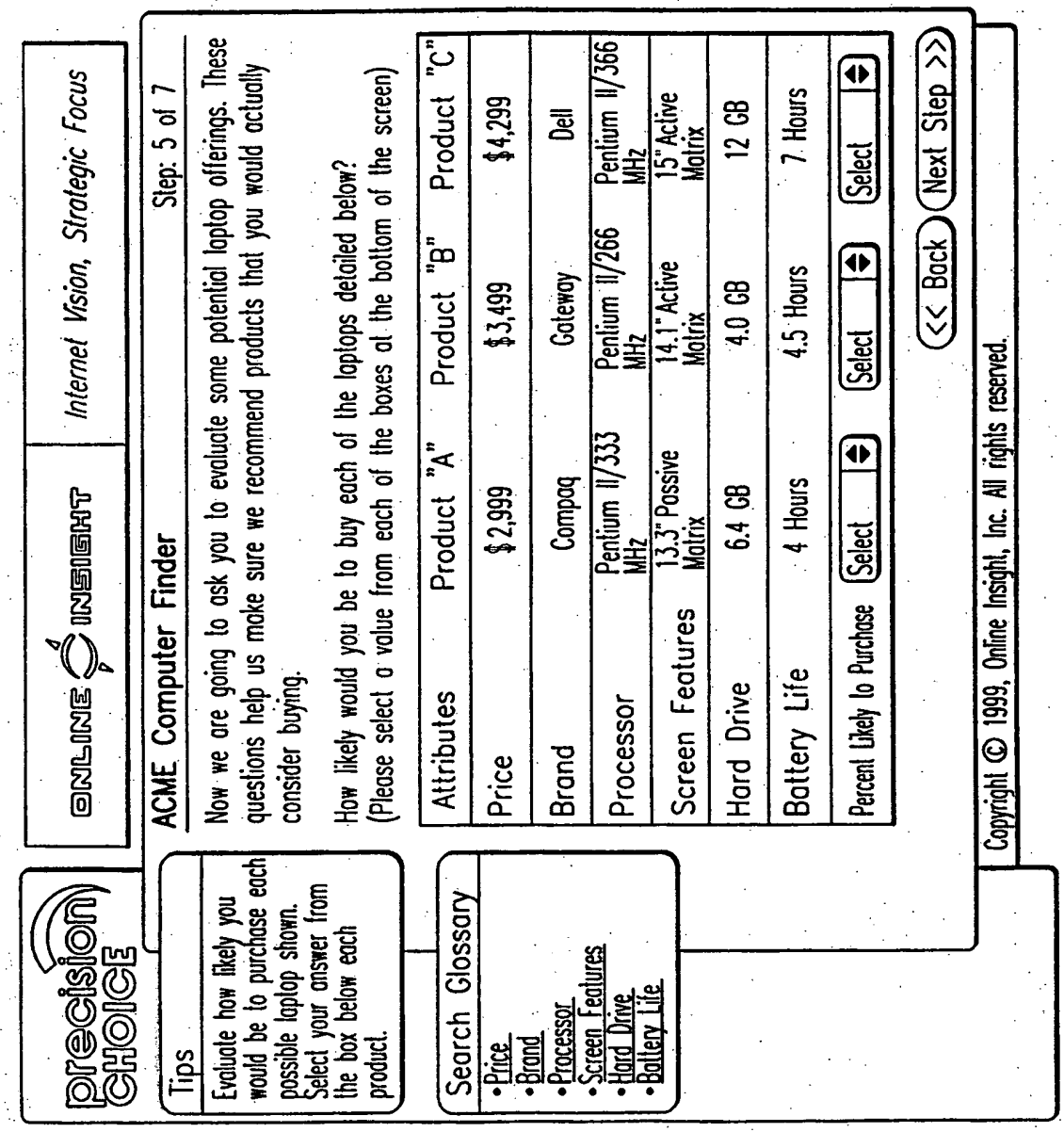

precision choice
ONLINE INSIGHT
Internet Vision, Strategic Focus
ACME Computer Finder
Step: 5 of 7
Tips
Evaluate how likely you would be to purchase each possible laptop shown. Select your answer from the box below each product.
Now we are going to ask you to evaluate some potential laptop offerings. These questions help us make sure we recommend products that you would actually consider buying.
How likely would you be to buy each of the laptops detailed below?
(Please select a value from each of the boxes at the bottom of the screen)
Search Glossary
• Price
• Brand
• Processor
• Screen Features
• Hard Drive
• Battery Life
Attributes
Product "A"
Product "B"
Product "C"
Price
$2,999
$3,499
$4,299
Brand
Compaq
Gateway
Dell
Processor
Pentium II/333 MHz
Pentium II/266 MHz
Pentium II/366 MHz
Screen Features
13.3" Possive Matrix
14.1" Active Matrix
15" Active Matrix
Hard Drive
6.4 GB
4.0 GB
12 GB
Battery Life
4 Hours
4.5 Hours
7 Hours
Percent Likely to Purchase
Select
Select
Select
<< Back
Next Step >>
Copyright © 1999, Online Insight, Inc. All rights reserved.

FIG. 11

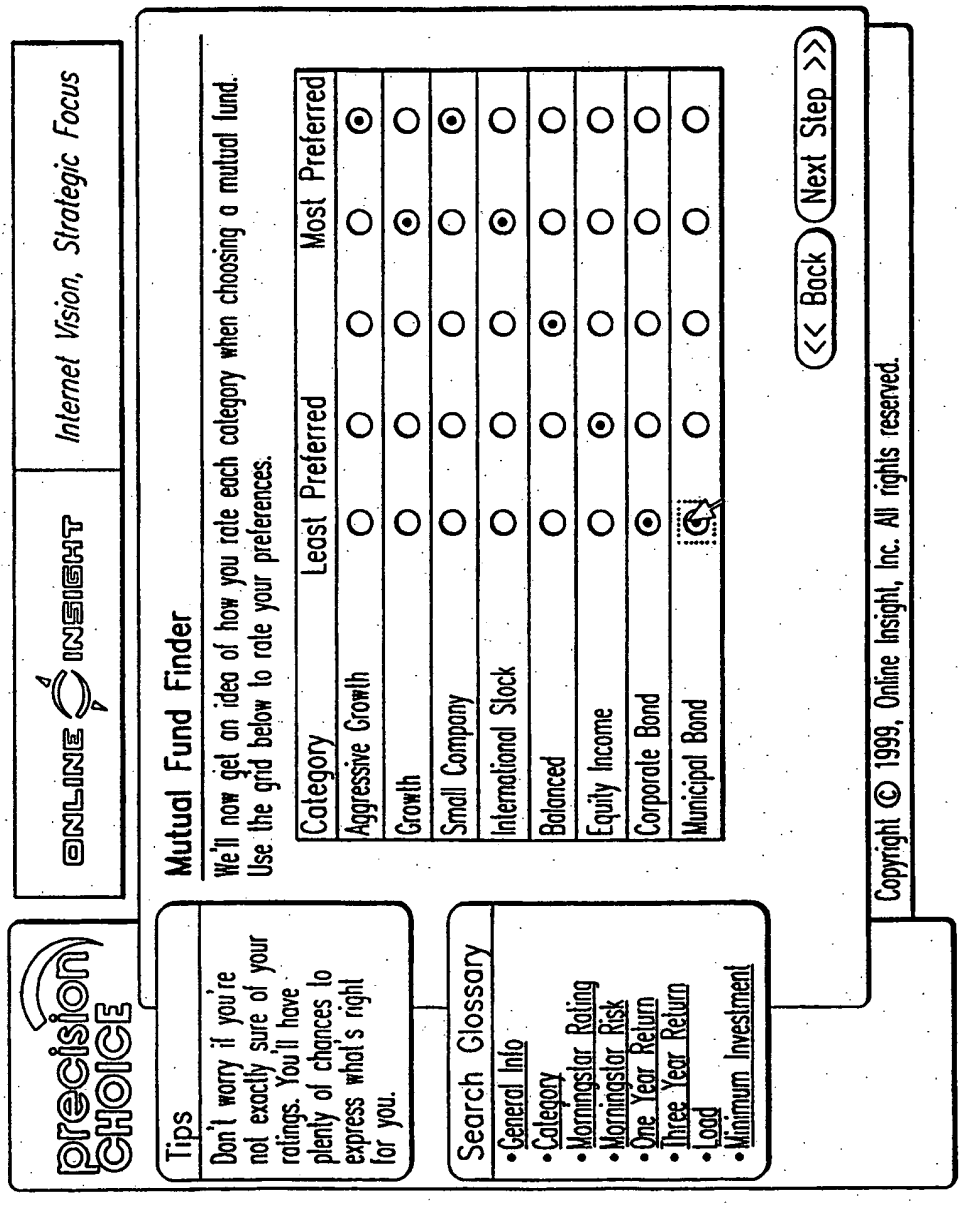

precision choice
ONLINE INSIGHT
Internet Vision, Strategic Focus
Mutual Fund Finder
We'll now get an idea of how you rate each category when choosing a mutual fund. Use the grid below to rate your preferences.
Category Least Preferred Most Preferred
Aggressive Growth
Growth
Small Company
International Stock
Balanced
Equity Income
Corporate Bond
Municipal Bond
<< Back
Next Step >>
Tips
Don't worry if you're not exactly sure of your ratings. You'll have plenty of chances to express what's right for you.
Search Glossary
General Info
Category
Morningstar Rating
Morningstar Risk
One Year Return
Three Year Return
Load
Minimum Investment
Copyright © 1999, Online Insight, Inc. All rights reserved.

FIG.12

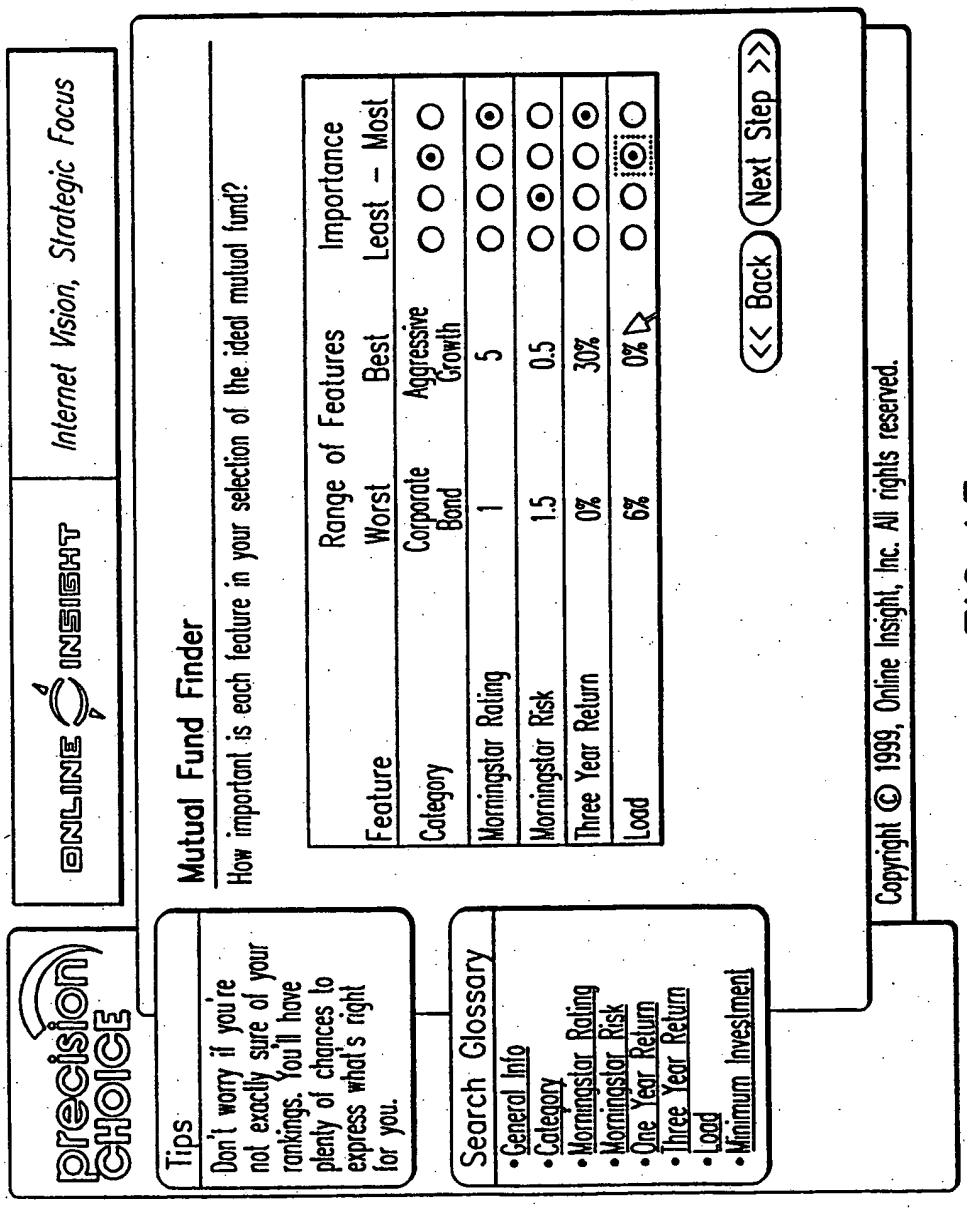

precision choice
ONLINE INSIGHT
Internet Vision, Strategic Focus
Mutual Fund Finder
How important is each feature in your selection of the ideal mutual fund?
Feature
Range of Features
Worst
Best
Importance
Least - Most
Category
Corporate Bond
Aggressive Growth
Morningstar Rating
1
5
Morningstar Risk
1.5
0.5
Three Year Return
0%
30%
Load
6%
0%
<< Back
Next Step >>
Tips
Don't worry if you're not exactly sure of your rankings. You'll have plenty of chances to express what's right for you.
Search Glossary
General Info
Category
Morningstar Rating
Morningstar Risk
One Year Return
Three Year Return
Load
Minimum Investment
Copyright © 1999, Online Insight, Inc. All rights reserved.

FIG. 13

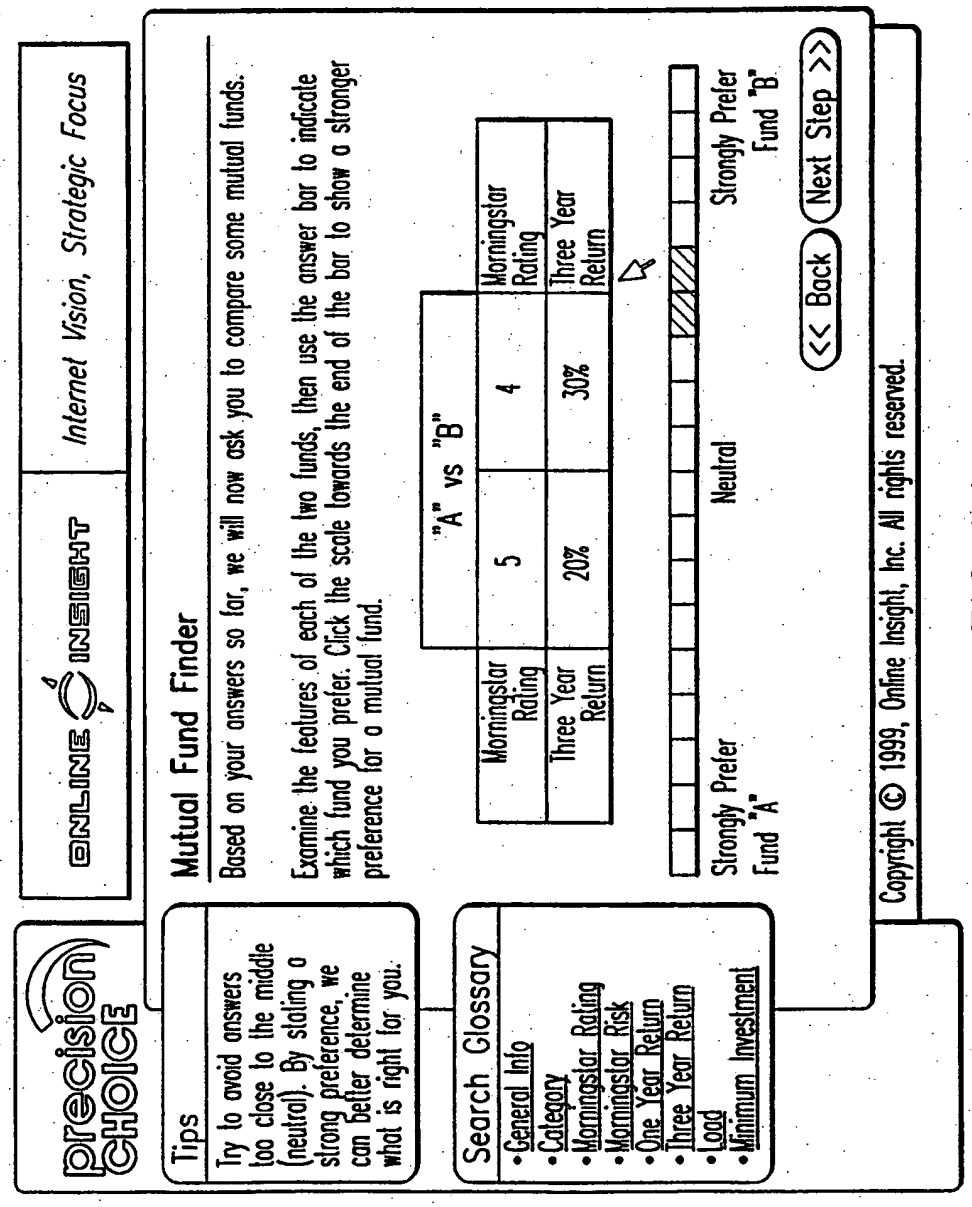

precision choice
ONLINE INSIGHT
Internet Vision, Strategic Focus
Tips
Try to avoid answers too close to the middle (neutral). By stating a strong preference, we can better determine what is right for you.
Search Glossary
General Info
Category
Morningstar Rating
Morningstar Risk
One Year Return
Three Year Return
Load
Minimum Investment
Mutual Fund Finder
Based on your answers so far, we will now ask you to compare some mutual funds.
Examine the features of each of the two funds, then use the answer bar to indicate which fund you prefer. Click the scale towards the end of the bar to show a stronger preference for a mutual fund.
"A" vs "B"
Morningstar Rating
5
4
Morningstar Rating
Three Year Return
20%
30%
Three Year Return
Strongly Prefer Fund "A"
Neutral
Strongly Prefer Fund "B"
<< Back
Next Step >>
Copyright © 1999, Online Insight, Inc. All rights reserved.

FIG.14

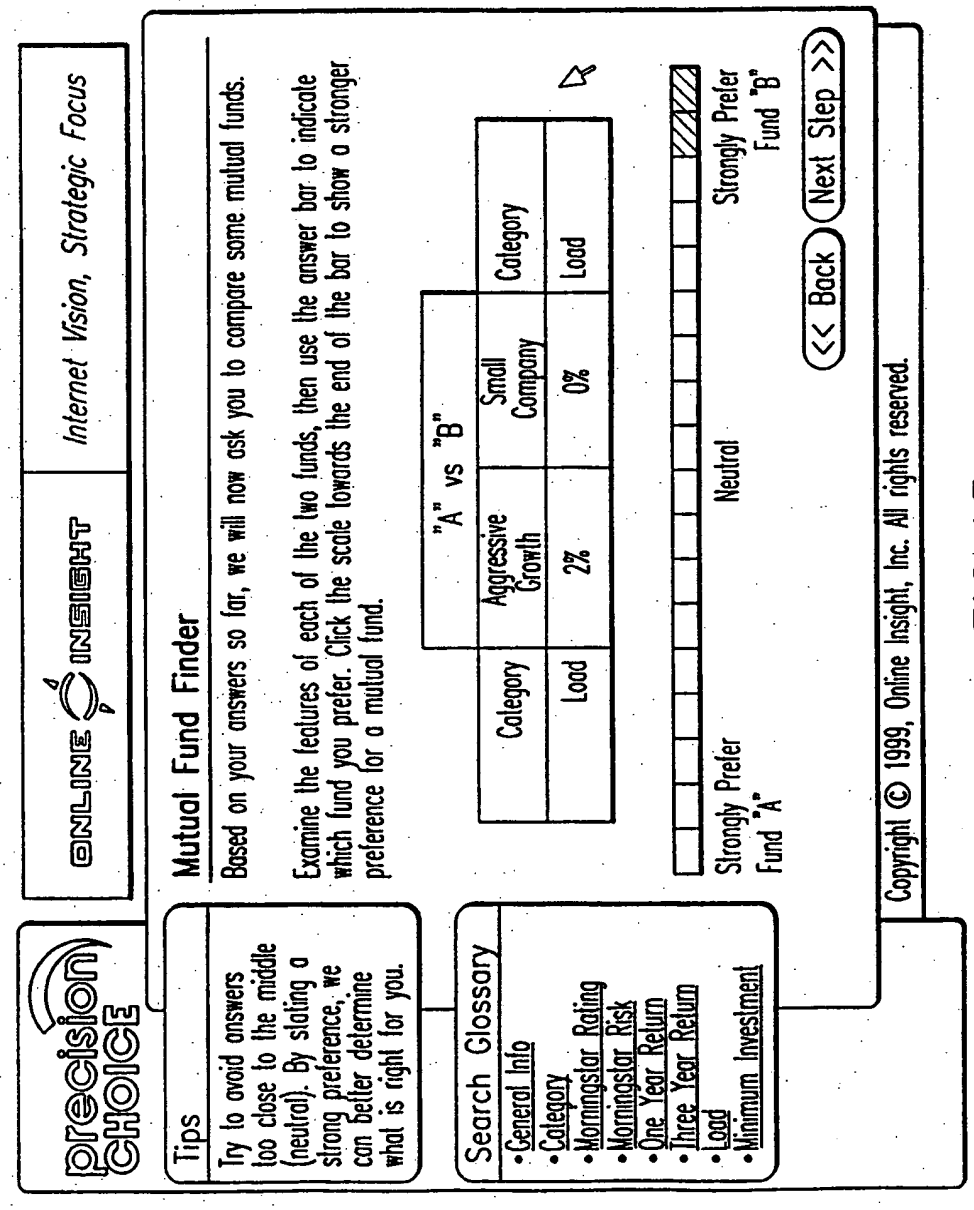

Precision Choice
Online Insight
Internet Vision, Strategic Focus
Mutual Fund Finder
Based on your answers so far, we will now ask you to compare some mutual funds.
Examine the features of each of the two funds, then use the answer bar to indicate which fund you prefer. Click the scale towards the end of the bar to show a stronger preference for a mutual fund.
"A" vs "B"
Category
Aggressive Growth
Small Company
Category
Load
2%
0%
Load
Strongly Prefer Fund "A"
Neutral
Strongly Prefer Fund "B"
<< Back
Next Step >>
Tips
Try to avoid answers too close to the middle (neutral). By stating a strong preference, we can better determine what is right for you.
Search Glossary
• General Info
• Category
• Morningstar Rating
• Morningstar Risk
• One Year Return
• Three Year Return
• Load
• Minimum Investment
Copyright © 1999, Online Insight, Inc. All rights reserved.

FIG.15

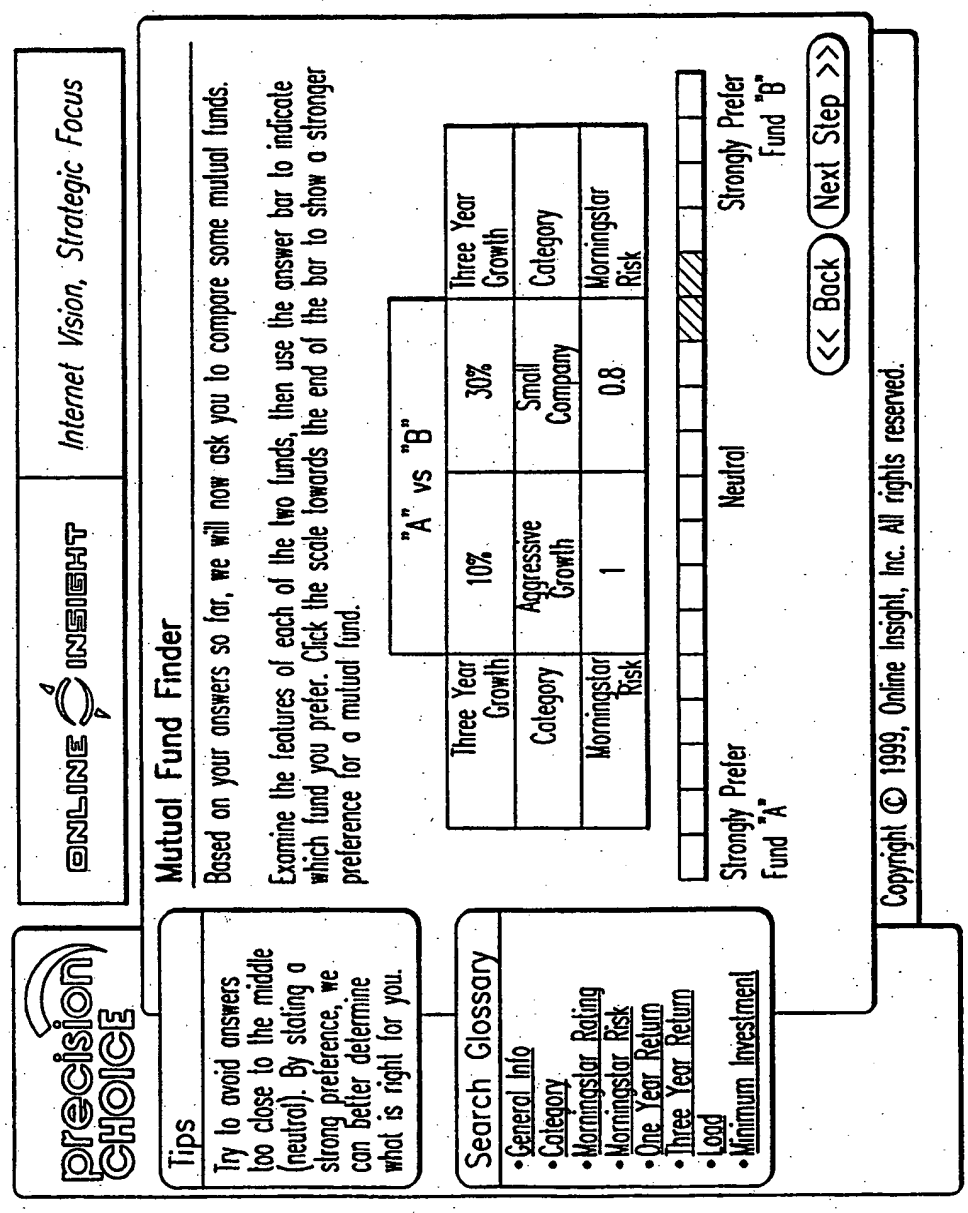

precision choice
ONLINE INSIGHT
Internet Vision, Strategic Focus
Mutual Fund Finder
Based on your answers so far, we will now ask you to compare some mutual funds.
Examine the features of each of the two funds, then use the answer bar to indicate which fund you prefer. Click the scale towards the end of the bar to show a stronger preference for a mutual fund.
"A" vs "B"
Three Year Growth | 10% | 30% | Three Year Growth
Category | Aggressive Growth | Small Company | Category
Morningstar Risk | 1 | 0.8 | Morningstar Risk
Strongly Prefer Fund "A"
Neutral
Strongly Prefer Fund "B"
<< Back
Next Step >>
Tips
Try to avoid answers too close to the middle (neutral). By stating a strong preference, we can better determine what is right for you.
Search Glossary
•General Info
•Category
•Morningstar Rating
•Morningstar Risk
•One Year Return
•Three Year Return
•Load
•Minimum Investment
Copyright © 1999, Online Insight, Inc. All rights reserved.

FIG. 16

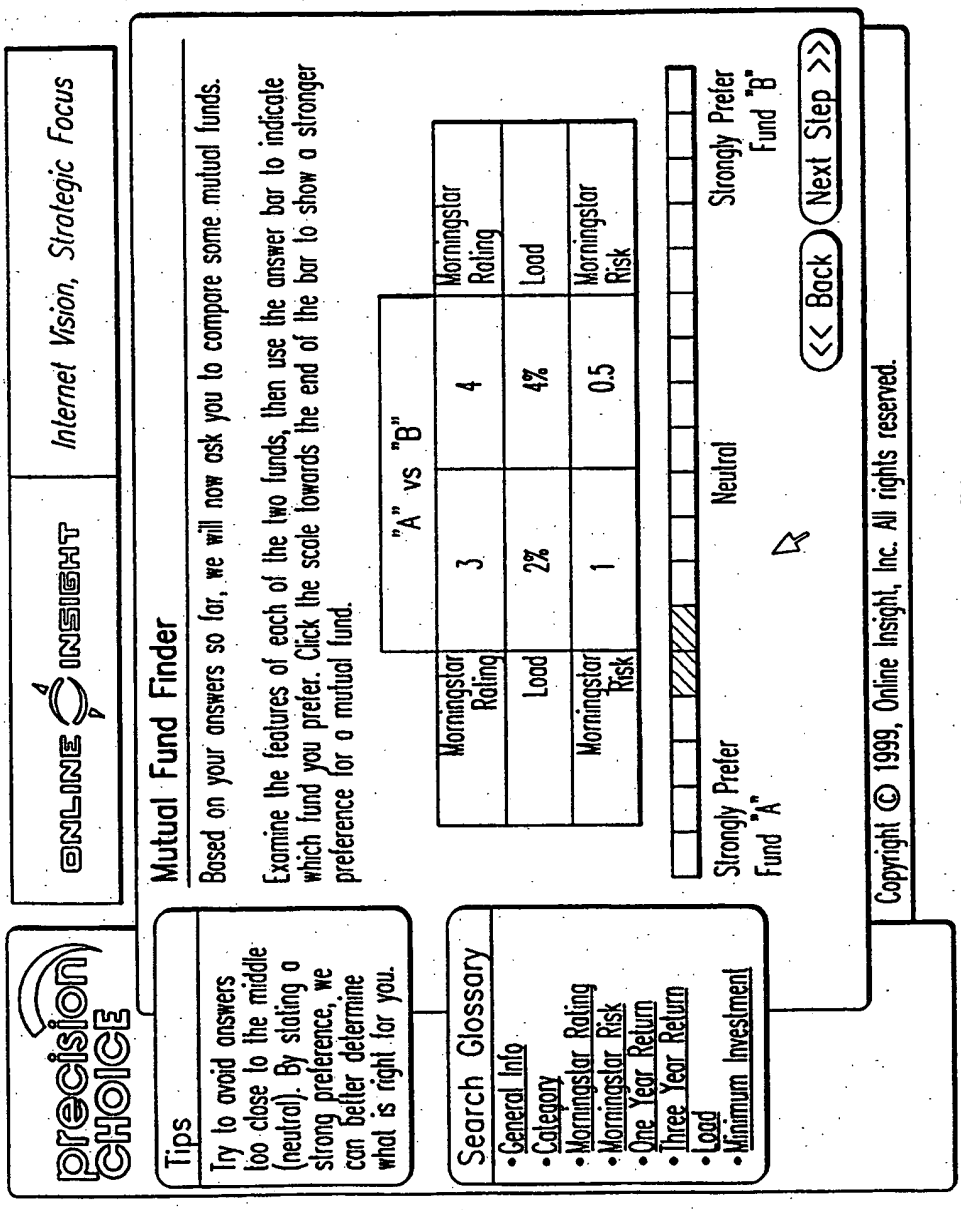

precision CHOICE
ONLINE INSIGHT
Internet Vision, Strategic Focus
Mutual Fund Finder
Based on your answers so far, we will now ask you to compare some mutual funds.
Examine the features of each of the two funds, then use the answer bar to indicate which fund you prefer. Click the scale towards the end of the bar to show a stronger preference for a mutual fund.
"A" vs "B"
Morningstar Rating 3 4 Morningstar Rating
Load 2% 4% Load
Morningstar Risk 1 0.5 Morningstar Risk
Strongly Prefer Fund "A"
Neutral
Strongly Prefer Fund "B"
<< Back
Next Step >>
Tips
Try to avoid answers too close to the middle (neutral). By stating a strong preference, we can better determine what is right for you.
Search Glossary
General Info
Category
Morningstar Rating
Morningstar Risk
One Year Return
Three Year Return
Load
Minimum Investment
Copyright © 1999, Online Insight, Inc. All rights reserved.

FIG. 17

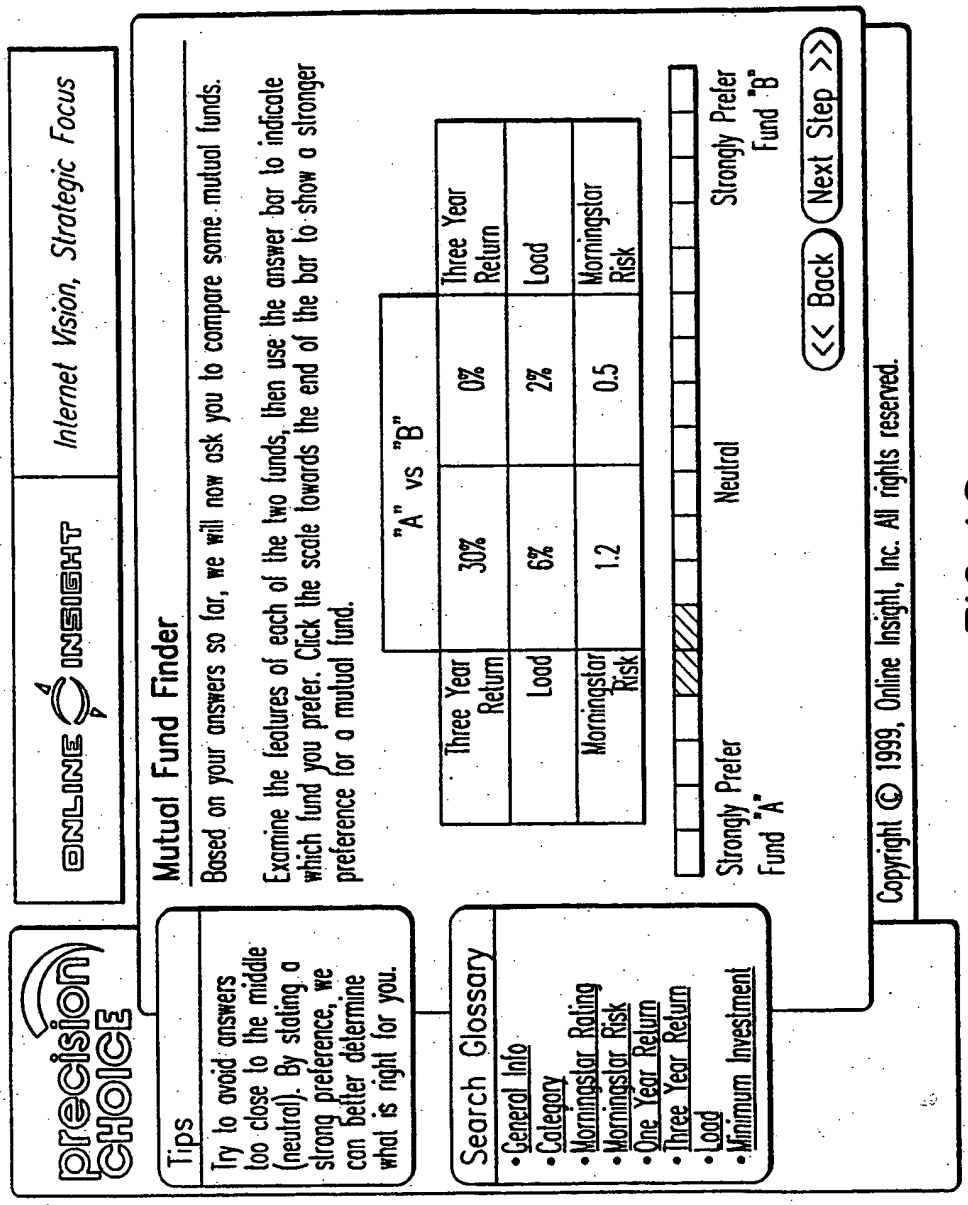

precision choice
ONLINE INSIGHT
Internet Vision, Strategic Focus
Mutual Fund Finder
Based on your answers so far, we will now ask you to compare some mutual funds.
Examine the features of each of the two funds, then use the answer bar to indicate which fund you prefer. Click the scale towards the end of the bar to show a stronger preference for a mutual fund.
"A" vs "B"
Three Year Return 30% 0% Three Year Return
Load 6% 2% Load
Morningstar Risk 1.2 0.5 Morningstar Risk
Strongly Prefer Fund "A"
Neutral
Strongly Prefer Fund "B"
<< Back
Next Step >>
Tips
Try to avoid answers too close to the middle (neutral). By stating a strong preference, we can better determine what is right for you.
Search Glossary
• General Info
• Category
• Morningstar Rating
• Morningstar Risk
• One Year Return
• Three Year Return
• Load
• Minimum Investment
Copyright © 1999, Online Insight, Inc. All rights reserved.

FIG. 18

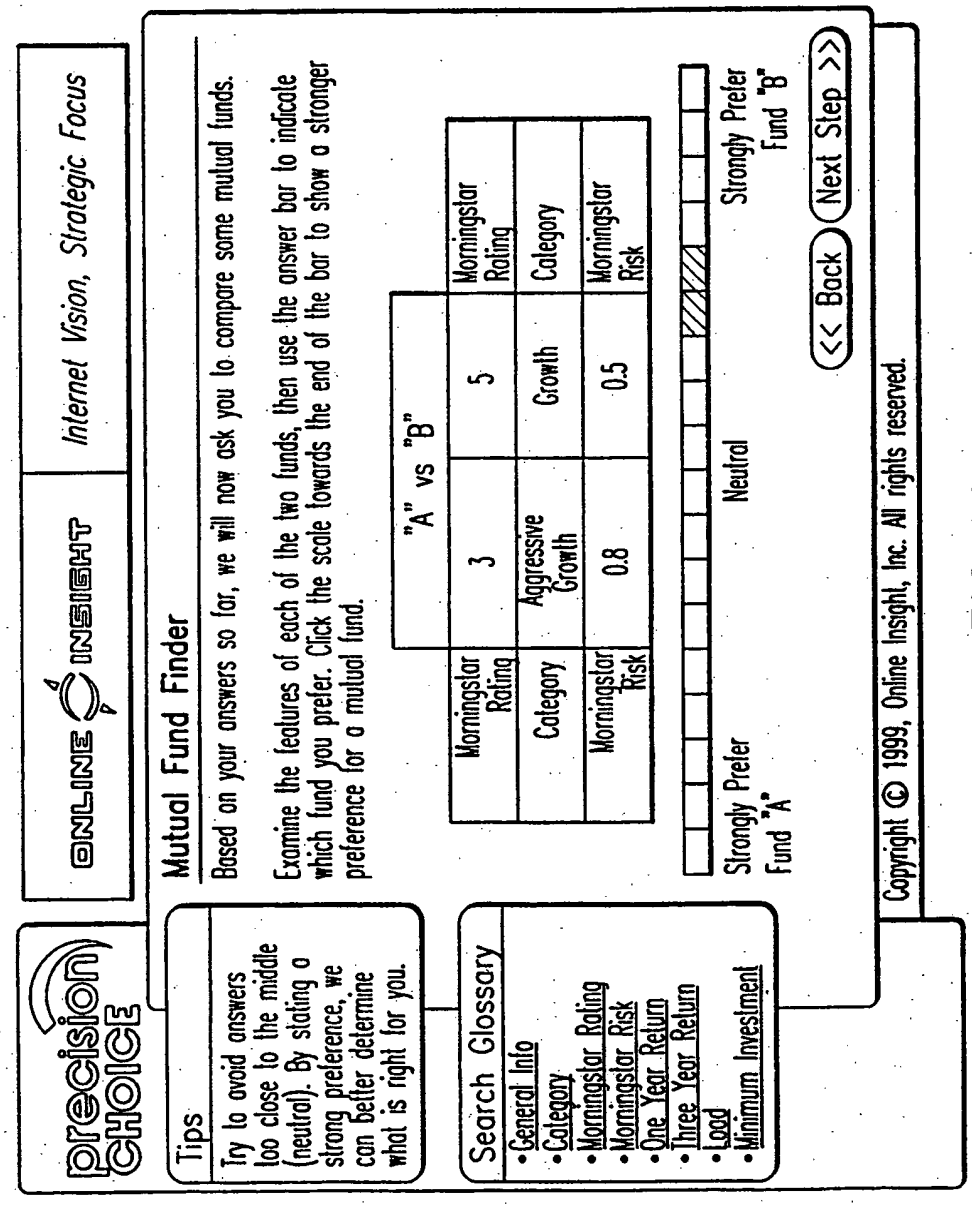

precision choice
ONLINE INSIGHT
Internet Vision, Strategic Focus
Tips
Try to avoid answers too close to the middle (neutral). By stating a strong preference, we can better determine what is right for you.
Search Glossary
•General Info
•Category
•Morningstar Rating
•Morningstar Risk
•One Year Return
•Three Year Return
•Load
•Minimum Investment
Mutual Fund Finder
Based on your answers so far, we will now ask you to compare some mutual funds.
Examine the features of each of the two funds, then use the answer bar to indicate which fund you prefer. Click the scale towards the end of the bar to show a stronger preference for a mutual fund.
"A" vs "B"
Morningstar Rating | 3 | 5 | Morningstar Rating
Category | Aggressive Growth | Growth | Category
Morningstar Risk | 0.8 | 0.5 | Morningstar Risk
Strongly Prefer Fund "A"
Neutral
Strongly Prefer Fund "B"
<< Back
Next Step >>
Copyright © 1999, Online Insight, Inc. All rights reserved.

FIG.19

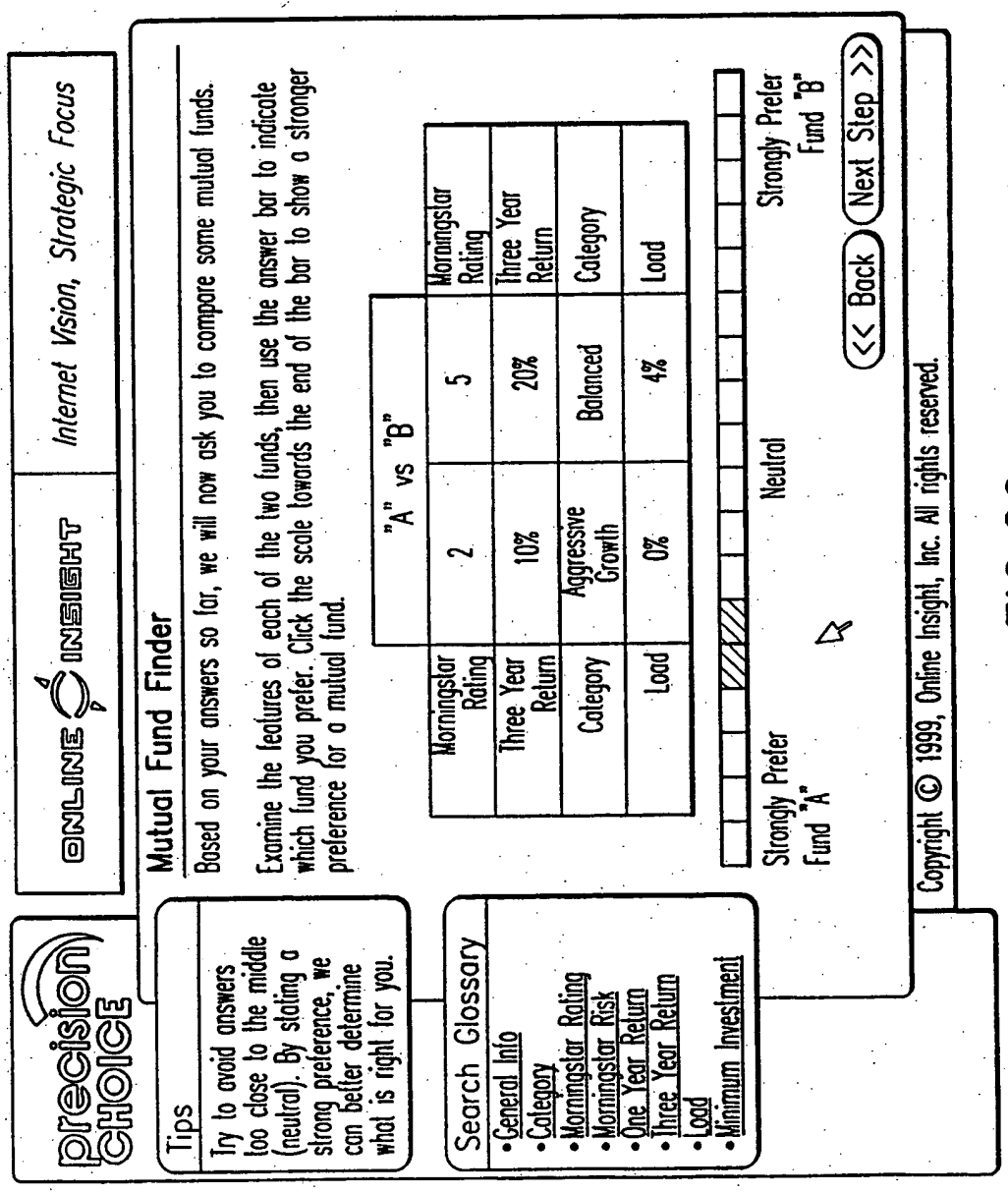

precision CHOICE
ONLINE INSIGHT
Internet Vision, Strategic Focus
Mutual Fund Finder
Based on your answers so far, we will now ask you to compare some mutual funds.
Examine the features of each of the two funds, then use the answer bar to indicate which fund you prefer. Click the scale towards the end of the bar to show a stronger preference for a mutual fund.
Tips
Try to avoid answers too close to the middle (neutral). By stating a strong preference, we can better determine what is right for you.
Search Glossary
General Info
Category
Morningstar Rating
Morningstar Risk
One Year Return
Three Year Return
Load
Minimum Investment
"A" vs "B"
Morningstar Rating 2 5 Morningstar Rating
Three Year Return 10% 20% Three Year Return
Category Aggressive Growth Balanced Category
Load 0% 4% Load
Strongly Prefer Fund "A"
Neutral
Strongly Prefer Fund "B"
<< Back
Next Step >>
Copyright © 1999, Online Insight, Inc. All rights reserved.

FIG.20

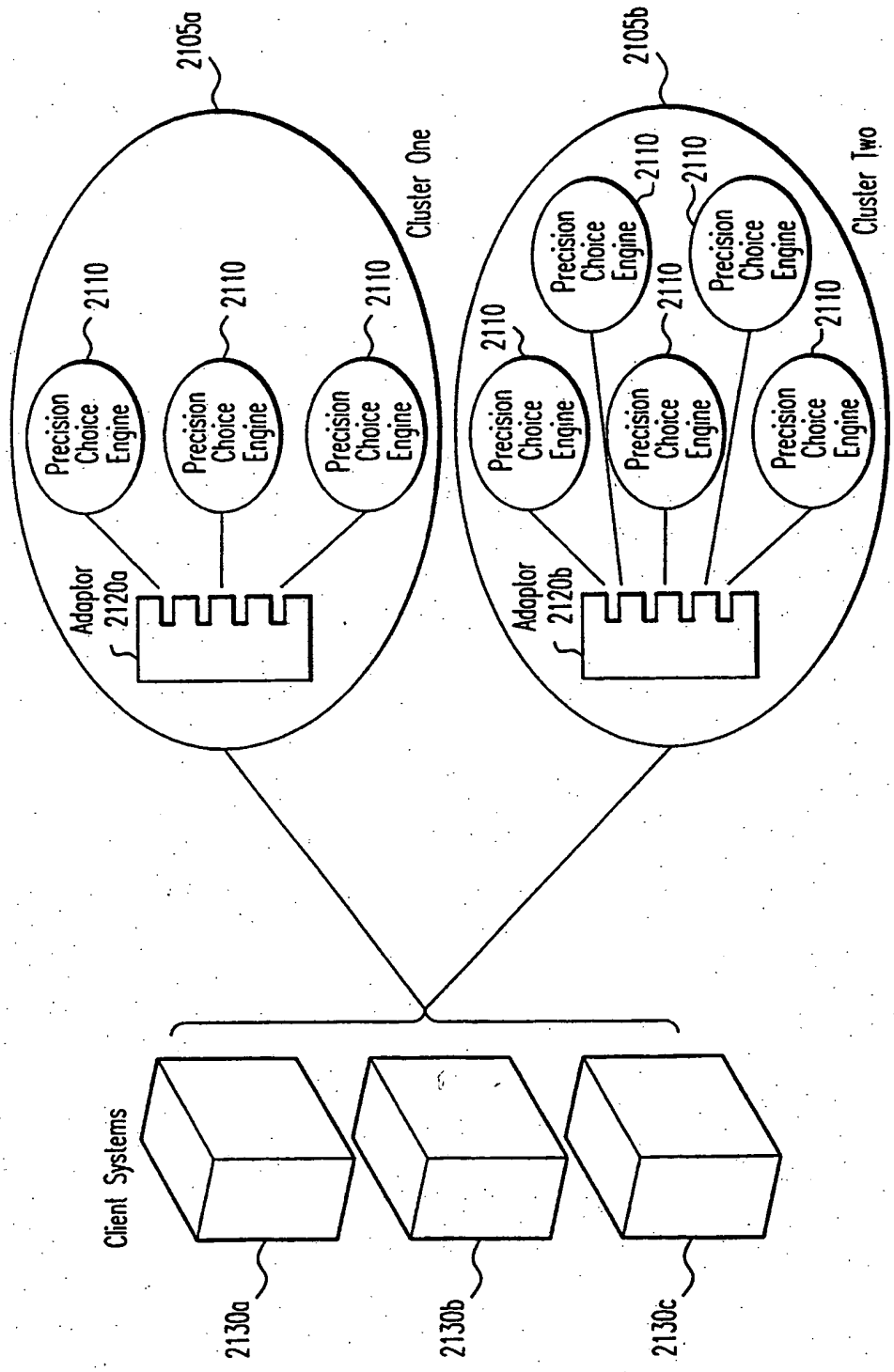

2105a
Cluster One
2110
Precision Choice Engine
Adaptor 2120a
2105b
Cluster Two
Client Systems
2130a
2130b
2130c

FIG.21

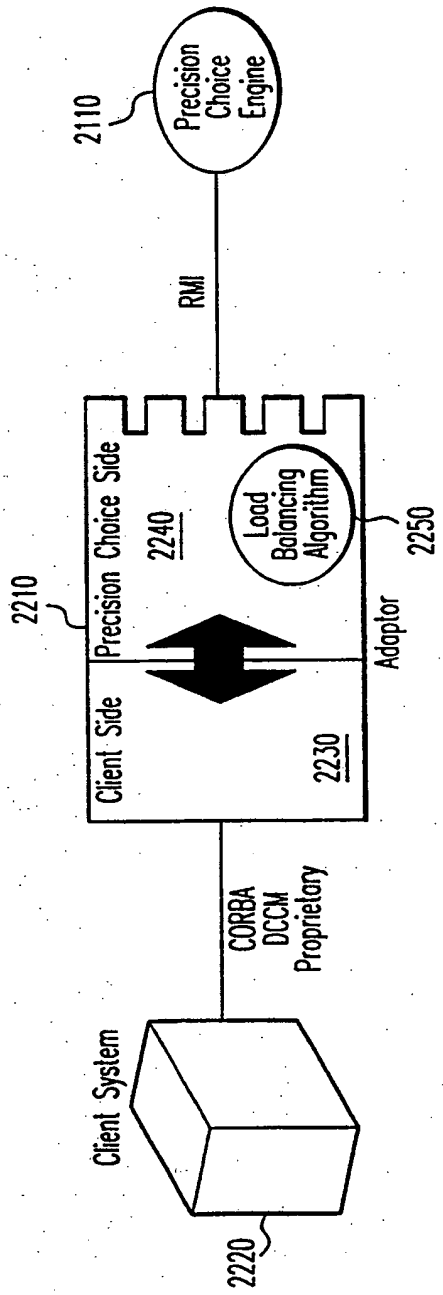

2110
Precision Choice Engine
RMI
2210
Precision Choice Side
2240
Load Balancing Algorithm
2250
Adaptor
Client Side
2230
CORBA DCCM Proprietary
Client System
2220

FIG.22

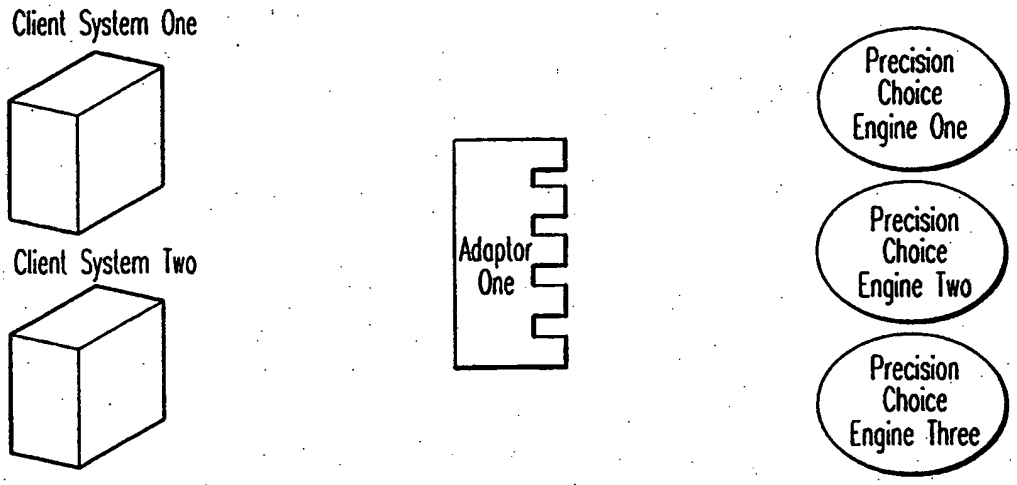

Client System One
Client System Two
Adaptor One
Precision Choice Engine One
Precision Choice Engine Two
Precision Choice Engine Three

FIG.23

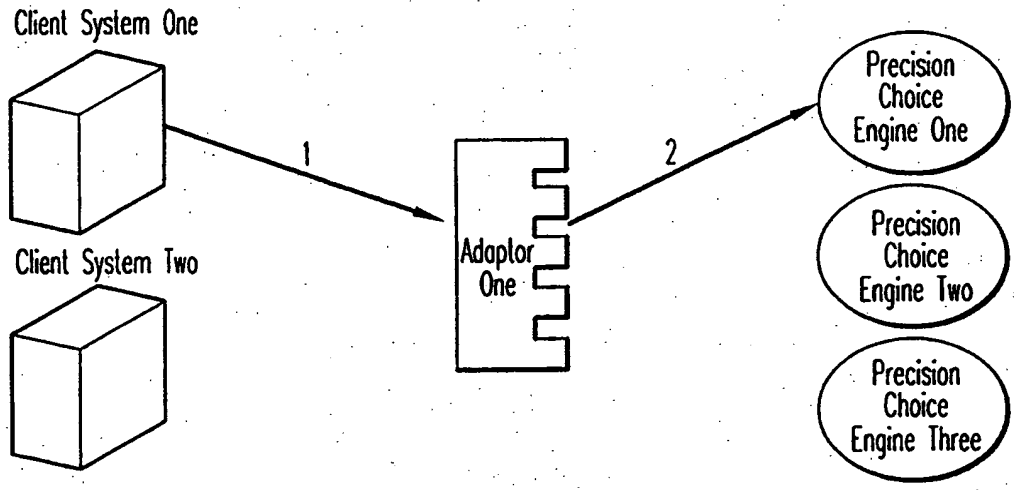

Client System One
Client System Two
1
2
Adaptor One
Precision Choice Engine One
Precision Choice Engine Two
Precision Choice Engine Three

FIG.24

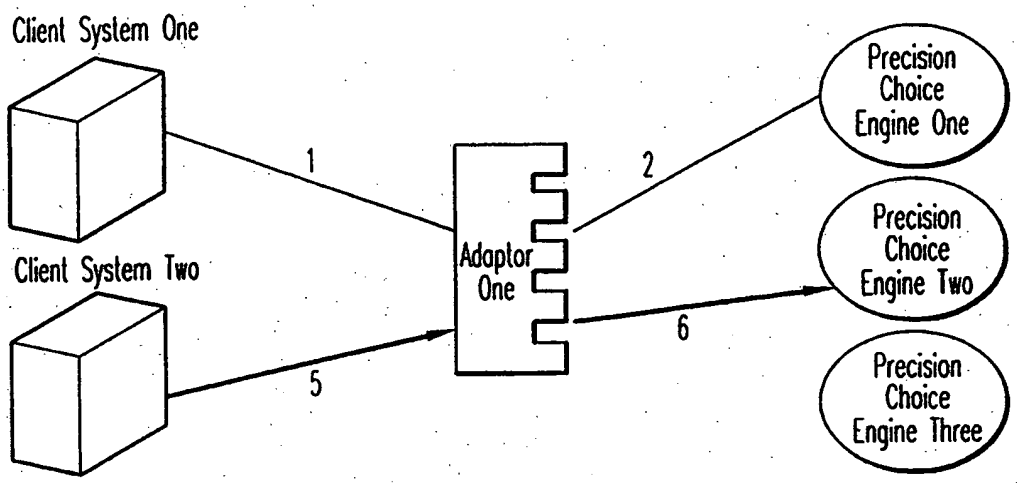

Client System One
Client System Two
1
2
5
6
Adaptor One
Precision Choice Engine One
Precision Choice Engine Two
Precision Choice Engine Three

FIG.25

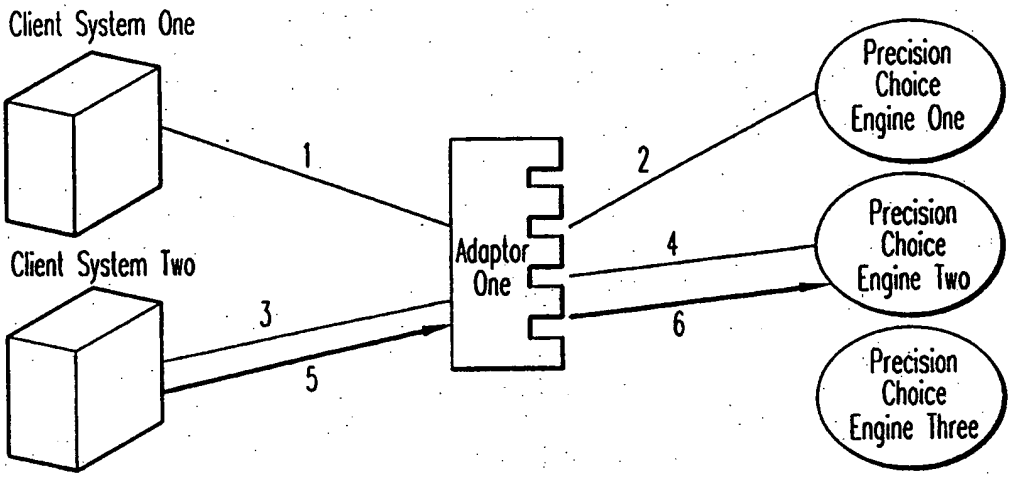

Client System One
Client System Two
1
2
3
4
5
6
Adaptor One
Precision Choice Engine One
Precision Choice Engine Two
Precision Choice Engine Three

FIG.26

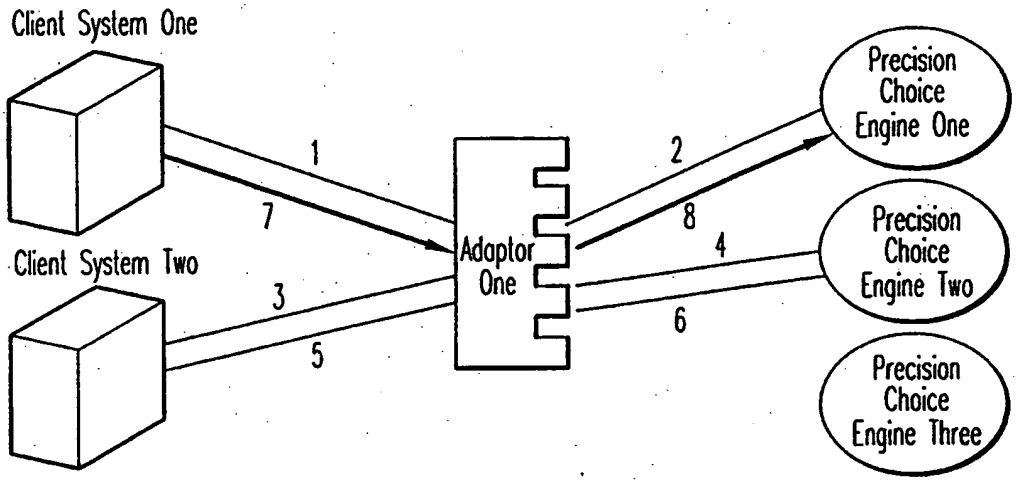

Client System One
Client System Two
Adaptor One
Precision Choice Engine One
Precision Choice Engine Two
Precision Choice Engine Three
1
2
3
4
5
6
7
8

FIG.27

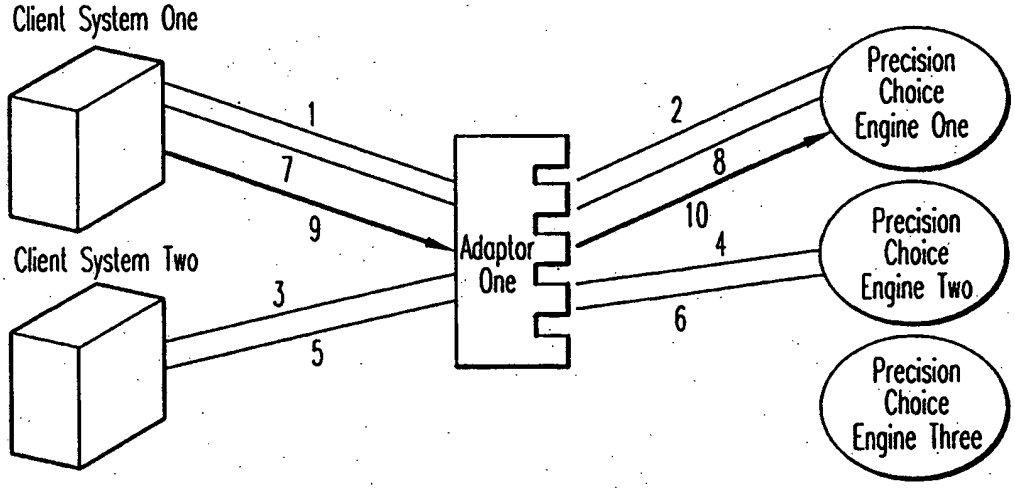

Client System One
Client System Two
Adaptor One
Precision Choice Engine One
Precision Choice Engine Two
Precision Choice Engine Three
1
2
3
4
5
6
7
8
9
10

FIG.28

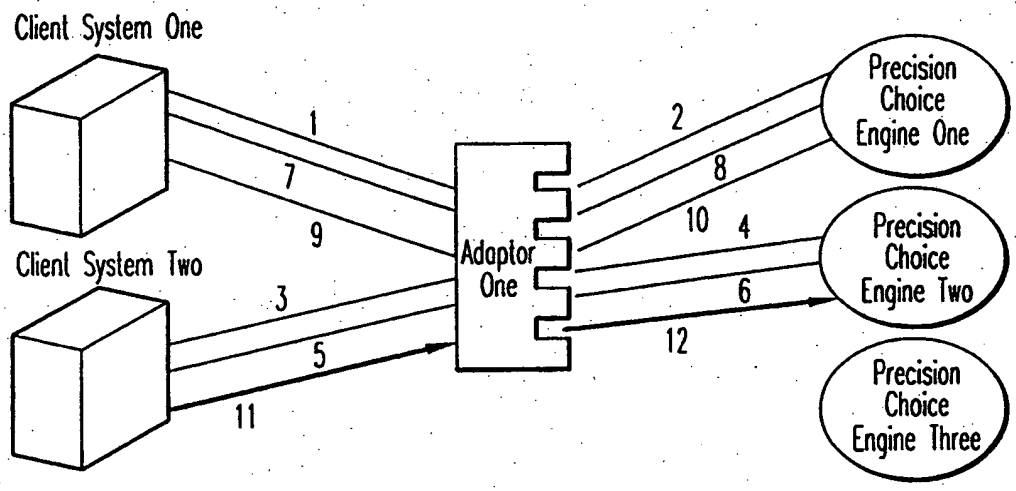

Client System One
Client System Two
Adaptor One
Precision Choice Engine One
Precision Choice Engine Two
Precision Choice Engine Three
1
2
3
4
5
6
7
8
9
10
11
12

FIG.29

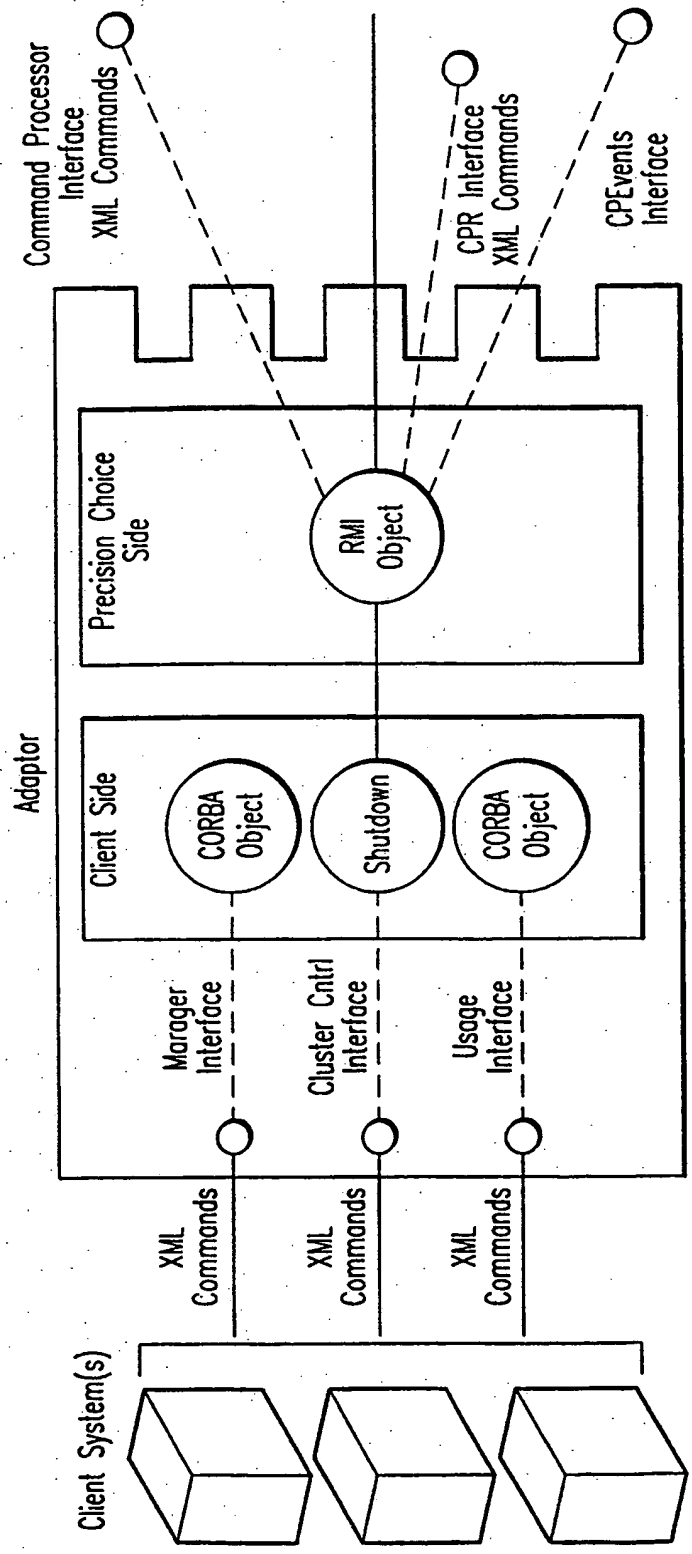

Command Processor Interface XML Commands
CPR Interface XML Commands
CPEvents Interface
Precision Choice Side
RMI Object
Adaptor
Client Side
CORBA Object
Shutdown
CORBA Object
Marager Interface
Cluster Cntrl Interface
Usage Interface
XML Commands
XML Commands
XML Commands
Client System(s)

FIG.30

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/02249

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) :G06F 17/60
US CL :705/10, 26, 27
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 705/10, 26, 27

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

NONE

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WEST, DIALOG

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 4,992,940 A (DWORKIN) 12 February 1991, col. 3, line 47 thru col. 10, line 53.	1-85
X	US 4,996,642 A (HEY) 26 February 1991, col. 3, line 63 thru col. 7, line 57.	1-85
X	US 5,717,865 A (STRATMANN) 10 February 1998, col. 3, line 35 thru col. 5, line 25.	1-85
X	US 5,749,081 A (WHITEIS) 05 May 1998, col. 3, line 10 thru col. 9, line 25.	1-85
X	US 5,790,426 A (ROBINSON) 04 AUGUST 1998, col. 6, line 56 thru col. 32, line 56.	1-85
X	SAATY. THOMAS L. Decision Making for Leaders, University of Pittsburg, RWS publications. 1988.	1-85

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 07 APRIL 2000	Date of mailing of the international search report 02 MAY 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer ALLEN MACDONALD James R. Matthews Telephone No. (703) 308-9708

Form PCT/ISA/210 (second sheet) (July 1998)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/02249

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X,P	US 5,884,282 A (ROBINSON) 16 March 1999, col. 6, line 56 thru col. 32, line 56.	1-85
A	US 5,124,911 A (SACK) 23 JUNE 1992, entire document.	1-85
A	US 5,041,972 A (FROST) 20 August 1991, entire document.	1-85
A	US 5,734,890 A (CASE et al) 31 March 1998, entire document.	1-85